INTEL CORPORATION
5200 N.E. Elam Young Parkway
Hillsboro, Oregon 97124-9987
(503) 681-8080

Dear Customer:

Thank you for purchasing Intel's UNIX Release ,4.0. Intel is pleased to be selling and supporting the industry standard UNIX for 386(TM) and i486(TM) based computers. Intel's UNIX Release 4.0 brings together System V, Xenix, and BSD systems.

Application binaries that run on UNIX Release 3.2 also run on UNIX Release 4.0. Most Xenix applications also run on Intel's Release 4.0. UNIX Release 4.0 from Intel provides many BSD/SunOS features and delivers a high degree of BSD/SunOS source level compatibility.

The product you've received includes a complete two user UNIX system, ILS support, Installation Guide, Release Notes, Product Overview and Master Index, and the following software:

* ST506/ESDI Boot Disk (5 1/4 and 3 1/2)
* WD SCSI Boot Disk (5 1/4 and 3 1/2)
* Adaptec SCSI Boot Disk (5 1/4 and 3 1/2)
* DPT EISA SCSI Boot Disk (5 1/4 and 3 1/2)
* Base System Disk 2 (5 1/4 and 3 1/2)
* Hardware Driver Set (5 1/4 and 3 1/2)
* UNIX 4.0 Installation Tape (60 MB)

Please call the appropriate number from the list below if any of these items are missing.

U.S.	800-INTEL4U
U.K.	(0) 793 / 641469

France	(16) 1 / 30.57.70.00
Germany	(0) 89 / 903-2025
Netherlands	(0) 10 / 4.07.11.30
Italy	(0) 2 / 892.00950
Israel	(0) 3 / 548-3222
Sweden	(0) 8 / 825416
Finland	(9) 0 / 544644
Spain	(9) 1 / 308-2552
Switzerland	call Germany
Denmark	call Sweden
Norway	call Sweden
Other countries	call nearest facility

A software product is only as good as the support behind it, especially a product as powerful and complex as the UNIX operating system. We have included our Installation Level Support (ILS) at no charge with your purchase. ILS is your assurance that you will be able to successfully install your Intel UNIX software.

ILS includes replacement of defective media within 90 days of purchase, one hour of installation support by phone, and access to Intel's UNIX bulletin board.

ILS also includes installation support for these related Intel UNIX software products:

Product Code	Description
UNX40ULTDLC	Multiuser Upgrade
UNX40OLDOC	On-line Documentation
MERGE40	DOS under UNIX capability
MOTIF40	Motif Graphical Interface

With ILS, support engineers will help you to get a shell prompt on the system console. ILS does not include network connection, application configuration, system administration, or application support.

Please call the appropriate number listed above to receive phone installation support. Hours of operation for U.S. and Canada are 8:00am to 5:00pm PST. Other locations operated during normal business hours. All local holidays are observed.

Write down your system's hardware configuration before you call - including attached hardware devices and add-in cards (manufacturer, model, and revision level are all helpful to the support engineer).

Fill out the product registration card, and mail it using the address label for your nearest Intel office.

By registering your products, you will receive bulletin board access, product update notices, and UNIX training information. Dial the bulletin board in the U.S. at (503) 640-3042. The bulletin board supports modems up to 2400 baud; you'll find the latest product information and various articles on the bulletin board.

Your initial session on the bulletin board is restricted to general public access, but you can request more access privileges. Log into the bulletin board, and choose the request menu to request upgraded privileges. Your access privileges will be upgraded by the next business day and will expire 30 days later.

Contact your Intel salesperson or Intel distributor for information on other Intel UNIX software products.

Thanks for choosing Intel.

intel

product release notes

UNIX® System V Release 4 Version 3

Release Notes

DOCUMENT 467693

SHEET 1 REV A

UNIX ® System V Release 4 Version 3 Release Notes
Order Number: 467693

Intel Corporation
3065 Bowers Avenue
Santa Clara, California 95052-8131

In locations outside the United States, obtain additional copies of Intel documentation by contacting your local Intel sales office.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products. (Registered trademarks are followed by ®.)

i486	Intel®	intel®	OpenNet
386	486	386	

TRADEMARKS

AT&T VDC 600 is a trademark of AT&T, Inc. AutoCAD is a registered trademark of AutoDESK, Inc. BSD is a registered trademark of the University of California, Berkeley, CA. Dell VGA is a trademark of Dell Computer Corporation, Inc. Everex is a trademark of Everex Systems, Inc. MS-DOS is a registered trademark of Microsoft Corporation, Inc. NFS is a registered trademark of Sun Microsystems, Inc. PostScript is a registered trademark of Adobe Systems, Inc. SunOS is a trademark of Sun Microsystems, Inc. UNIX is a registered trademark of UNIX System Laboratories, Inc, in the U.S. and other countries. IBM and PC AT are registered trademarks of International Business Machines Corporation. PC XT is a trademark of International Business Machines Corporation. MDS is an ordering code only and is not used as a product name or trademark. MDS is a registered trademark of Mohawk Data Sciences Corporation. Soft-Scope is a registered trademark of Concurrent Sciences, Inc. XENIX, MS-DOS, and Microsoft are trademarks of Microsoft Corporation. Ethernet is a registered trademark of Xerox Corporation. Hazeltine and Executive 80 are trademarks of Hazeltine Corporation. TeleVideo is a trademark of TeleVideo Systems Inc. Wyse and WY-75 are registered trademarks of Wyse Technology. MIX is an acronym for Modular Interface eXtension and is a registered trademark of MIX Software, Incorporated.

PREFACE

This document contains important performance, bug and workaround information on UNIX System V Release 4 Version 3. The document reflects input and modification requests (MR's) received at the time of release from both USL developers and USL customers. The following notes are organized into eight chapters reflecting eight generalized sections of our product. Additional information on feature enhancements, performance and programming environment changes is also provided in these chapters. Customers needing more in-depth reference information must consult the Master Index and the appropriate reference manuals and user's guides in the UNIX System V Release 4 documentation set.

CONTENTS

CHAPTER 1. INSTALLATION NOTES

1.1 BEFORE INSTALLING

The following sections contains information needed before installing UNIX System V Release 4 (SVR4) on your system.

1.1.1 ESDI Disk Controllers

With SVR4, large disks with more than 1024 cylinders may appear smaller than they are. This is because SVR4 does not recognize more than 1024 cylinders (for compatibility with other operating systems).

On ESDI disk controllers that provide transparent logical translation, enabling the logical translation will overcome this problem.

1.1.2 Cartridge Tape Controllers

The SVR4 cartridge tape package (for non-SCSI tape controllers) currently supports the following controller/drive combinations:

TABLE 1. Cartridge Tape Controllers and Drives

Tape Controller	Supported Tape Drive
Wangtek PC-36	Wangtek 5099-EN
Wangtek PC-36	Wangtek 5125-EN
Wangtek PC-36	Wangtek 5150-EN/EQ
Everex 811 (B.Tech)	Wangtek 5150-EN/EQ
Archive SC499-R	Archive Ext. FT-60
Archive VP402	Archive Viper 2150L
Everex 811 (B.Tech)	Archive Viper 2150L
Archive VP402	Wangtek 5150-EN/EQ

To install the reference binary from tape, the controller hardware must have the following settings:

Base address for I/O port	0x300
Interrupt vector	5
DMA channel	1

— NOTE —

See your hardware cartridge tape reference manual for instructions on how to set the base address, interrupt and DMA to those listed above.

If you do not have a tape controller or your controller is not configured correctly, the following message is displayed at boot time:

```
Cartridge Controller was not found at address 00000300H
Tape driver disabled
```

The cartridge tape will be inoperative.

1.1.3 Boot Diskettes

Boot disks are now write-protected. This is normal; do not remove the write-protect tabs.

1.1.4 Size of Swap

When running networking applications, the size of `/dev/swap` should be at least 11MB, or equal to 2 times the size of installed memory, whichever is larger. You can allocate this space at installation time or use the `swap -a` command to add additional space if your system is already installed.

1.1.5 Mixing Packages

Packages in filestream format (pre-SVR4) should not be mixed with packages in datastream format during installation. Install all of one type of package, then quit out of the installation script and install the other type. All the SVR4 software packages are in datastream format.

1.1.6 Disk Partitioning with DOS/UNIX

The DOS 4.01 `fdisk` program assumes it can store diagnostic information in the last cylinder available on the hard disk, so if a UNIX partition is created that uses the last cylinder (MS-DOS 4.01) `fdisk` will be unable to create a DOS partition.

The workaround is to either create the UNIX partition at the front of the disk so the last cylinder isn't used, or create the DOS partition using the UNIX `fdisk` (not DOS `fdisk`) and never delete the DOS partition.

If you boot from a DOS floppy, you will not be able to access a physical DOS partition on the hard disk, drive C, unless it is the first partition created by `fdisk`.

1.1.7 tar Compatibility

The SVR4 `tar` command will not restore pre-SVR4 `tar` archives properly if either of the following conditions are true:

- The archive contains directories (versus regular files), or
- a file within the archive spans a volume boundary.

When either of these conditions are true, the pre-SVR4 version of the `tar` command can be used to restore the archive. The suggested workaround is to save and rename the command. After installing SVR4, install the old tar command on your system.

1.1.8 Mouse Support

Only Logitech mice are supported.

1.1.9 Problem with /tmp Filesystem When System Boots

If the hard disk is configured with `/tmp` as a separate filesystem, `/tmp` is not cleared out when the system boots.

1.1.10 Unnamed Slice Offered as Default When Configuring 2nd Hard Disk

When partitioning a second hard disk, after selecting the desired optional slices, a recommended disk configuration (the recommended size for each slice) displays. The recommended configuration may include an unnamed slice of 2 or 3 cylinders.

- If you accept the configuration shown, the unnamed slice will be created on the second hard disk and you lose the 2 or 3 MB of disk space.
- If you do not want the unnamed slice to be created, reject the configuration shown. You will be asked to specify the size of each selected slice (in cylinders). You won't be asked about the unnamed slice. Specify the desired sizes. When again asked if the configuration is acceptable, say `y`.

See Chapter 2 and Appendix B in the *System V Release 4 Installation Guide* for details on partitioning the hard disk and selecting optional slices.

1.2 AFTER INSTALLING

1.2.1 Booting Problems

In rare instances, when you reboot the system after installing the SVR4 software, the reboot fails. This may happen because the system is trying to reclaim shadowed (aliased) RAM memory with methods which are not compatible with your hardware.

If your system fails to reboot, do the following corrective procedure.

— NOTE —

This procedure may also be used to install the original default configuration file over one that may be corrupted.

1. Put the first boot diskette into disk drive 1 and press the RESET button. The system boots.
2. When prompted to insert the second boot diskette, remove the first diskette and insert the second one.
3. Type **`Newconfig`**.

   If your system is the standard machine type, a new copy of the default configuration file is installed. Go to Step 5 to continue.

   If your system is a non-standard machine type, the system asks:

   ```
   Do you want to reclaim the BIOS SHADOW RAM?
   ```

   - If you type **`y`** (which is the default), a new copy of the default configuration file for the machine type is installed.
   - If you type **`n`**, a new configuration file that does not reclaim shadow RAM is installed.
4. Answer as desired.
5. When prompted, remove the second boot diskette then press **< Ctrl-Alt-Del>** . The system will reboot and your system should be ready to use. If the system still does not boot, contact your customer service representative.

CHAPTER 2. OPERATING SYSTEM

2.1 GENERAL NOTES

2.1.1 Performance

UNIX System VR4 Version 3 is a substantially better performing system than Version 2 on machines with more than 4 megabytes of memory.

This is primarily due to fixes to UFS to eliminate unnecessary disk writes and improved tuning (instituted automatically in the installation scripts). These enhancements bring about a 10% to 50% system improvement on most configurations. Systems with four megabytes of memory will see performance comparable to Version 2 since their performance is primarily determined by memory considerations.

Version 3 performance on industry benchmarks is generally equal to or better than Version 2.

2.1.2 Security

The kernel has been changed to re-initialize all per-process resource limits to the system defaults provided in the `mtune` file whenever a set-user-ID or set-group-ID file is `exec`'d. These limits include both the new BSD-style resource limits [see `getrlimit(2)`] and pre-SVR4 resource limits [see `ulimit(2)`]. In all other cases, per-process resource limits are inherited across `exec` as they have been in the past. This change prevents unprivileged processes from causing privileged processes to fail by depriving them of necessary resources.

2.1.3 Unsupported Executable Types

XENIX non-separate Instruction & Data binaries such as AUTOCAD Release 10 are not supported. The exec code fails and attempts to run the binary as a shell script.

2.1.4 Service Access Facility

Under the new Service Access Facility, tty ports are now monitored by `ttymon`, a STREAMS-based tty port monitor that can monitor one or more ports. Most of the `getty` and `uugetty` entries in `/etc/inittab` are converted to equivalent entries under `ttymon`. `/etc/gettydefs` is augmented by a new formatted file, `/etc/ttydefs`. For compatibility, `getty` and `uugetty` are still provided and are required for use on non-STREAMS drivers. They may not, however, be provided in future releases.

If the `getty/uugetty` configurations in the pre-SVR4 `/etc/inittab` are desired, a command, `ttyconv` may be used to convert all `getty/uugetty` entries in `/etc/inittab` to equivalent setups under `ttymon`. The `ttyconv` command also converts `/etc/gettydefs` to `/etc/ttydefs`, if the `/etc/gettydefs` file exists on the system. The conversion should be done in single user or administration mode as there are many changes to `/etc/inittab` entries. Direct copying of the old `/etc/inittab` file over the new one is not recommended.

2.1.5 Source Compatibility Issues

2.1.5.1 Header Files Header file enhancements in SVR4 may cause compilation errors in pre-SVR4 applications. In a small number of cases, header file names may need to be changed, or new header files added to pre-SVR4 program source code.

Header Files and POSIX Type Definitions. The convergence of standards in SVR4 has introduced a number of new names into header files. It is inevitable that these new names may cause compilation errors. The simplest solution to these errors is to rename conflicting identifiers.

Table 2 includes some of the new POSIX type definitions which have been introduced into SVR4.

TABLE 2. New POSIX Type Definitions

Defined Types	Description
`gid_t`	Used for group IDs
`mode_t`	Used for some file attributes
`nlink_t`	Used for link counts.
`off_t`	Used for file sizes.
`pid_t`	Used for process IDs and process group IDs.
`uid_t`	Used for user IDs.

2.1.6 Expanded Fundamental Types (EFT)

This section presents a brief discussion of the expanded size of the fundamental operating system types (EFT) in SVR4 and their possible impact on future application code development. SVR4 provides for the expansion of certain defined kernel data types which were formerly defined as 2-byte types. Included in Table 2 is the list of expanded fundamental data types in SVR4.

The resource demands placed on the UNIX operating system by implementation on larger machines, together with the advent of distributed networks and the evolution of the operating system, has contributed to defined data types exceeding their original design capacities of two bytes. For example, some UNIX implementations have worked around the shortage of minor device numbers by assigning multiple major numbers to a single driver. This has generated a need to expand the limits on these data types. The EFT feature in SVR4 will answer this need.

2.1.6.1 The EFT Feature In the kernel and in "new" drivers, the definition for certain UNIX System fundamental data types will be four bytes instead of the two or one as in pre-SVR4 systems. We call these defined data types "fundamental" because their use pervades most of the UNIX System.

The expansion allows for a period of transition to ensure compatibility for the installed base of UNIX Systems. Therefore, in SVR4 the expansion has been designed to offer coexistence between the existing base of software and the present and future need to provide for the support of larger configurations. In SVR4, the type expansion provides, in most cases, forward application source and binary compatibility for pre-SVR4 systems. Source and object compatibility will be supported in SVR4 for those drivers written to the AT&T Device Driver Interface (DDI) specification. STREAMS-based drivers, however, will have to be recompiled. It is important to mention that software affected by EFT will have to be modified to include the POSIX and System V defined `typedefs` in Table 2 before values for the EFT data types grow beyond the 2-byte boundary. See the section *Some General Pointers for Applications Development* for exceptions.

Table 3 lists the fundamental data types affected by the EFT expansion along with their underlying type. The "New Name" column indicates whether the symbolic type name (e.g. `uid_t`) is new in SVR4.

TABLE 3. Basic EFT Data Types

Type	Name	Underlying Type	New Name
uid	`uid_t`	`long`	yes
gid	`gid_t`	`long`	yes
pid	`pid_t`	`long`	yes
ino	`ino_t`	`ulong`	no
dev	`dev_t`	`ulong`	no
mode	`mode_t`	`ulong`	yes
nlink	`nlink_t`	`ulong`	yes
major	`major_t`	`ulong`	yes
minor	`minor_t`	`ulong`	yes

These types will, by default, be four bytes in any application that uses them.

As mentioned earlier, however, this does not mean that applications will fail in SVR4. For example, if an application were to pass the `chown` system call a *uid* and *gid* of type `int` (or even `short`), everything will still work because the arguments are automatically promoted. Also, `stat` will continue to work even if you copy a *value* from the `stat` structure into a `short` or an `int` because of automatic demotion. The reason this works is that, even though the *sizes* of the fields are larger in SVR4, the *values* are being kept within their pre-SVR4 limits. For example, the `stat` structure can now hold a *uid* of over 2 billion, but to maintain compatibility with existing binary applications, the maximum value for a *uid* will fit in 2-bytes for SVR4. See the section *Some General Pointers for Applications Development* for exceptions to this rule.

These restrictions on values will be maintained for at least SVR4, but will be relaxed beyond SVR4 (and may be relaxed in certain binary releases, as decided by the source licensee building the binary). Applications are thus strongly urged to start using the new symbolic types. For example, every declaration of a *uid* should be changed from `int` or `short` to `uid_t`. This will ensure that when these limits are relaxed in the future, application compatibility will not be affected.

2.1.7 Some General Pointers for Applications Development

In SVR4, although forward `a.out` compatibility is supported, there are certain conditions where "`.o`" and source compatibility problems may arise. The following conditions should serve as general caveats for applications as well as for kernel functions:

- The linking of EFT and non-EFT object (`.o`) files that pass EFT structures, or reference EFT structures by address may result in a corrupted `a.out` file. This incompatibility is assumed to be small as it deviates from portable coding standards. Since the values for EFT data types will remain small in SVR4, parameter passing by value will be unaffected with the exception of `dev_t` (see below).

- An incompatibility will arise in source code that references EFT data types by address (pointers) when their underlying C type has a different size than the object it references. For example, the `st_uid` field in `stat` is a 2-byte integral type in pre-SVR4 systems. For EFT, this field is a 4-byte integral type. Applications that previously used a short-pointer to reference `st_uid` will break when compiled in a SVR4 environment. Although the above coding example is not considered portable, to work in SVR4 the pointer declaration for a `uid` object must be to type `uid_t`.

  User level source code affected by EFT can defer source migration in SVR4 by defining the feature test macro `_STYPES` to obtain the pre-SVR4 system definitions (where fundamental data types will be small and pre-SVR4 interfaces used). Affected kernel source will have to be changed to work in SVR4.

- Although automatic promotion and demotion of arguments is available in SVR4 (for maximum portability during the transition period), formatted data types are not subject to this process. For example, the `st_dev` data type includes two components, a major and minor device number. In SVR4, the major device number occupies the upper 14 bits and minor number lower 18 bits of `dev_t`. If the device number is assigned to a variable two bytes wide, then the device number will be truncated. This is the only case in EFT where pass-by-value will be affected when the receiving variable is two bytes wide. This incompatibility is assumed to be small since pre-SVR4 source code should already include `dev_t` declarations.

- The accounting data file has been expanded in SVR4 to support EFT data types. The accounting flag `ac_flag` has been set to `AEXPND` to indicate a SVR4 account file. This may cause potential incompatibility since some software has dependencies on this file. This incompatibility is small and affects mostly administrative programs.

- The archive format for `cpio` changed in SVR4 to support EFT. In SVR4, the `cpio` header will default to EFT. Consequently, `cpio` archives that will be restored on pre-SVR4 systems should be created with the `-H odc` option [see `cpio`(1) and `cpio`(4)].

- MAXMINOR is a tunable parameter. Its default is the same as it is in UNIX System V Release 3.2.

— NOTE —

The implementation of the EFT feature uses static function definitions within header files such `<sys/stat.h>`. Programs that include such headers and that are compiled on pre-SVR4 systems may no longer compile successfully on SVR4 systems if one of the affected headers is included within the body of the function. Furthermore, the inclusion of static functions in header files causes a problem for programs that depend on the line number information in program objects. These programs include `sdb`, `lprof`, and `dis`.

2.1.8 STREAMS

St_size Field in stat Structure. The interpretation of the `st_size` field of the `stat` structure changes in SVR4. Since a pipe in System V Release 4 is bi-directional, there are two separate data flows. Therefore, the size `st_size` returned by a call to `fstat`, where the argument is the file descriptor of either end of the pipe, is the number of bytes available for reading from that file descriptor. Previously, the size `st_size` returned by a call to `fstat`, where the argument was the write-end of a pipe, was the number of bytes available from the read end.

STRMSGSZ. Any code written for STREAMS drivers that expected `STRMSGSZ` to be greater than 0 should note that 0 is now a legal value for `STRMSGSZ`.

Flush Handling. Flush handling has been expanded in STREAMS to include flushing up to 256 individual bands of data flow. Users of pre-SVR4 modules should be aware that a flush message to these modules will flush all of the queues of data.

RFS. The semantics of the STREAMS `ioctl` system calls on named streams over RFS are not supported entirely. Also note that modules pushed on a named stream over RFS from a remote machine may not give the expected results.

TTY. The terminal subsystem in SVR4 has been converted to utilize the STREAMS-based character I/O mechanism. All the base system drivers including console, asynchronous ports, XT and SXT have been converted to STREAMS. The `clist` character I/O mechanism, however, has been maintained for compatibility. It is strongly recommended that all drivers be converted to STREAMS.

The SVR4 terminal subsystem provides SVR3.x, POSIX, BSD and XENIX compatibility. SVR3.x and POSIX functionality has been included in the standard terminal line discipline (`ldterm`) and the drivers. There is some BSD functionality such as in `word erase`, `reprint`, `echo control` that is part of the standard line discipline. All of these BSD enhancements are controlled using the flag `IEXTEN` (which is turned off by default).

The BSD and XENIX terminal `ioctl` compatibility is provided by pushing a specialized line discipline (`ttcompat`) on the top of the standard line discipline. While `ttcompat` allows most BSD and XENIX terminal applications to be run on SVR4, it is recommended that the applications be converted to POSIX `termios` or `termio` interface to follow the future direction. BSD applications should use `/usr/ucbinclude/sys/ioctl.h` instead of `/usr/include/sys/ioctl.h`. All other BSD terminal related files are in `/usr/include/sys`.

ptys. Along with hardware STREAMS drivers, SVR4 provides a STREAMS-based pseudo-terminal subsystem (`ptys`). `ptys` are useful for remote login and windowing types of applications.

2.1.8.1 Source Compatibility Source compatibility has been maintained for STREAMS code. Note, however, that with the enabling of EFT, much information has been moved and has changed size. The STREAMS source code has also been re-organized into "public" and "private" pieces. The header file `sys/stream.h` contains the public data structures that modules or drivers may need to reference. The header file `sys/strsubr.h` contains data structures private to the STREAMS subsystem, which should not be referenced by drivers or modules.

2.1.9 Virtual Memory

2.1.9.1 Swap Space Configuration A SVR4 system may need more swap space (an increase bounded by the size of real memory) to run a load that ran under SVR3. SVR3 attempted to use the sum of real memory and swap space for what SVR4 calls anonymous memory (memory without real file backup). The SVR3 method had some design holes that could cause a deadlock, but it did allow configurations with large amounts of real memory and small amounts of swap space to run loads whose use of anonymous memory exceeded the available swap space. Under SVR4, swap space is required for all anonymous memory (which avoids the deadlock issues of SVR3). The SVR3 deadlock problems result from pages having both a swap and a real memory association after swap while only being counted once. The SVR4 approach avoids this by needing swap space for all anonymous memory.

2.1.10 File System Independent Booting

The BFS file system type does not support the following features:

mmap	truncate up
symbolic links	volcopy
labelit	mkdir

2.1.11 Signals - Source Compatibility

The table below lists the new signal types in SVR4.

TABLE 4. New Signals in SVR4

Signals	Processes
SIGSTOP	job control
SIGTSP	
SIGCONT	
SIGTTIN	
SIGTTOU	
SIGVTALARM	real time
SIGPROF	
SIGXCPU	rlimit
SIGXFSZ	

To prevent these signals from being sent to processes unprepared to accept them, processes ignore these signals, by default. Processes must explicitly change that disposition in order to receive them.

In the future, `init` will not cause its children to inherit a `SIG_IGN` disposition for these new signals.

2.1.12 kdb Package Installation

When installing the `kdb` package, if you use the `pkgchk` command, the message:

```
No pathnames associated with this package.
```

displays. Ignore the message.

2.2 X WINDOWS NOTES

2.2.1 Initial Setup

If you do not explicitly set the `XDISPLAY` environment variable in your startup script (e.g. `.profile`), then you will need to do the following:

1. Type **`/usr/X/adm/oladduser`**
2. Type **`. .profile`**

This will set your `XDISPLAY` variable when you login and put `/usr/X/bin` in your path.

2.2.2 Set Up Instructions

`xterm` has OPEN LOOK[1] features; cut-and-paste is handled differently than in previous releases. Please read the *OPEN LOOK User's Guide* carefully for details.

In particular, the user may notice that CTRL-c, CTRL-v, CTRL-p and other keystrokes are not interpreted as expected. *xterm*, as a part of its OPEN LOOK Look and Feel, interprets these keystrokes as shortcuts for some OPEN LOOK commands. To get around this (which `vi` users may desire, and `emacs` users have to do), change the properties either using `olwsm` (WARNING: this program WILL alter your `.Xdefaults` file) or change the `.Xdefaults` file manually.

To use the second method, edit the `.Xdefaults` file, and add the following lines to your `.Xdefaults` file:

```
*copyKey:       Mod1< c>
*cutKey:        Mod1< x>
*pasteKey:      Mod1< v>
*propertiesKey: Mod1< p>
*stopKey:       Mod1< s>
*undoKey:       Mod1< u>
```

The keystrokes are mapped from their control key combination to a counterpart which uses the ALT key,

For those of you who have old X clients, you need to watch out for one major change from X11R2 to X11R3—fonts. The locations for accessing fonts has changed, and the actual

1. OPEN LOOK is a trademark of AT&T

fonts available may be different than before. If an old client dies with an X error, look carefully at the error message to determine if a previously available font is either no longer available, or if the font is available via another name.

Experienced users with their own *xinitrc* files should make sure that the file contains one client that will "stay around" for the duration of the session. Do NOT have all clients put in the background.

2.2.3 xterm Command

2.2.3.1 xterm does not always startup iconically `xterm` with the `-i` option does not always startup the xterm iconically.

2.2.3.2 Passwd command does not work in xterm If you type **passwd** while in an `xterm`, the system responds:

```
Usage:
passwd [-s] [name]
```

To change your password on an X window, you must type:

`passwd` *your-login-name*

To work around this problem, you can switch to the console via a vt switch (by pressing **<ALT-SYSREQ> <H>**), then execute the `passwd` command. To get back to the `xterm`, press **<ALT-SYSREQ> <P>**.

2.2.4 Olfm Does Not Change Directory

After searching into a sub-directory and performing a menu entry from a file selection, the command that is performed from the `.olfmrc` file is not done in the current directory. The current working directory remains in the directory that `olfm` was started in.

2.2.5 Known Problems

1. The client `xwd` has been known to fail on occasion when a window is selected for "dumping". If this happens, the use may wish to use either `xwininfo` or `xlswins` to determine the numeric window id for the desired window. The numeric window id may then be passed to `xwd` using the `-id` option or use `olprintscreen`.
2. Old (pre-X11 Release 3) clients often worked in prior releases of X despite failing to follow some explicit requirements of the X protocol. If you run an old client and see an error message like "`X Protocol error detected by server: integer parameter out of range...`", the client may contain an instance of the bug. The client should still be usable if you invoke the server with the "backward (or bug) compatibility" option. To do this, start *xinit* with the *bc* option.

For example,

```
xinit -- bc
```

3. Users should be sure to provide some `sleep` time between invocations of successive clients in their `.xinitrc` or `.olinitrc` file. Attempting to start many clients at once could result in clients never showing up on the display.

4. `xpic`[2] will report that no fonts are available.

   To have access to fonts, place the following lines in a file called `.xpic` in the user's home directory:

```
default Times-Roman 10
Times-Roman *-times-medium-r-normal-*
Times-Italic *-times-medium-i-normal-*
Times-Bold *-times-bold-r-normal-*
Times-BoldItalic *-times-bold-i-normal-*
Courier *-courier-medium-r-normal-*
Courier-Bold *-courier-bold-r-normal-*
Courier-BoldOblique *-courier-bold-o-normal-*
Courier-Oblique *-courier-medium-o-normal-*
Helvetica *-helvetica-medium-r-normal-*
Helvetica-Bold *-helvetica-bold-r-normal-*
Helvetica-BoldOblique *-helvetica-bold-r-normal-*
Helvetica-Oblique *-helvetica-medium-r-normal-*
Symbol *-symbol-medium-r-normal-*
Special *-symbol-medium-r-normal-*
Roman *-times-medium-r-normal-*
Bold *-times-bold-r-normal-*
BigBold *-times-bold-r-normal-*
Image-Roman *-times-medium-r-normal-*
Image-Italic *-times-medium-i-normal-*
```

5. `xpic` is known to have problems deleting splines and some lines.

6. There is an odd interaction between some fonts and `xterm`. Extraneous bits may be left at the bottom of a line. The user may not even notice the problem, depending on the fonts used and the resolution of the screen.

---

2. Copyright University of Toronto 1988, 1989. Written by Mark Moraes

7. `xmj`[3] installs its own colormap. This can be distracting when entering and leaving the game's window. No harm should result from this, however.

8. The game `roids`[4] starts up with a very small window, and it is too slow to be useful. The user may wish to remove it.

9. `xcalendar`[5] does not know the correct location for its help file. To work around the problem, edit the application defaults file (`/usr/X/lib/app-defaults/XCalendar`) so it references the correct help file. Add the line:

```
XCalendar*helpFile: /usr/X/lib/xcalendar.hlp
```

10. Some DOS applications have been known to set the mouse in an odd state (whether run under a DOS emulator, or run under DOS before booting off the UNIX partition). If the mouse behaves incorrectly when X is started, fix the problem by disconnecting then reconnecting the mouse.

11. Can't cut and paste between the new `xterm` and X11R2 clients. To workaround, paste into an `xedit`(1) window and save the text to a file, then display the file in an xterm anad cut from there.

2.3 SPECIFIC NOTES

2.3.1 /stand

The system will only boot from a kernel that is in `/stand`. If a kernel pathname is given that is in a directory other than `/stand`, the directory path part of the filename will be removed, and `/stand` will be searched for the named kernel (i.e. if a user types `/etc/conf/cf.d/unix.mine`, what they get is `/stand/unix.mine`).

2.3.2 ksh Fails to Interrupt

The following commands typed interactively to `ksh` will not allow the user to `INTR` out of `while`.

---

3. Original source Copyright 1988 Exceptions
   X11 portions Copyright 1989 Concurrent Computer Corporation
4. Copyright 1989 Digital Equipment Corporation
5. Copyright 1988 Massachusetts Institute of Technology

```
ksh$ ls -i | while read file
> do
> sleep 10
> done
<del>  <---------  INTR keyboard character.
```

`ksh` will continue until the script is completed. Also, the command `kill 0` doesn't work.

2.3.3 ksh Behavior and MAXUP Process Limits

If you attempt to fork more than `MAXUP` processes, the shell will wait until your processes are complete before giving you a prompt.

2.3.4 init Behavior When Executed Inside a vt

In UNIX System V Release 4 you may run administrative commands such as the `init(1M)` command on terminals other than the integral keyboard/display. Never issue the `init s` command from within a Virtual Terminal (VT) on the integral console to bring the system down to "Single User Mode". If you do, two shell processes will attempt to read from your keyboard simultaneously, causing unpredictable keyboard behavior and the confusion of your input.

If this problem occurs on your terminal, do the following workaround:

When prompted to enter the maintenance (`root`) password, press < CTRL-D> (hold down CTRL and press D) several times until you see the # prompt. You should then be able to administer your system in the single user mode.

2.3.5 Debugging and vt Devices

If the kernel debugger is entered manually on an AT (by pressing < CTRL-ALT-D>) while the active terminal is a `vt` (i.e. not `/dev/console`), the debugger will be started but the monitor will not be switched from the vt session to the console, leading to the appearance of a hung system. Typing <q> <CR> on the home console will quit the debugger and resume operation of the system. Therefore, make sure you are on the real console (`tty` returns `/dev/console`) before entering the debugger.

2.3.6 Cartridge Tape

On a SCSI-based system, SCSI cartridge tape software (`st01`), and the integral cartridge tape driver (`qt`) use the same device nodes in `/dev/rmt` which causes `idmknod` errors.

The workaround involves changing the device name entries. After installing the cartridge tape add-on package, change the device name entries for either `st01` or `qt` by editing the node file in `/etc/conf/node.d`. The device names are the second entry in each

line of the node file.

For example:

```
rmt/c0s0 ⟶ rmt/C0s0
```

After making the changes, create the new device nodes by typing `/etc/conf/bin/idmknod`.

2.3.7 Tape Capacity Limitation

When using a high-density tape drive (SCSI or non-SCSI) such as the Archive Viper 2150 L or S, and the Wangtek 5150EN or ES, use the Archive 660 or 3M DC600XTD tape cartridges. Otherwise, you will only be able to write 125 MB of data on a 150 MB tape.

2.3.8 Simultaneous Cartridge Tape and Floppy use

Systems utilizing the Western Digital WD1007 Combo Disk Controller may experience floppy read/write failures when the floppy access occurs during a data transfer to or from an integral 1/4" cartridge tape unit.

2.3.9 Recompilation of Graphics Applications

The `ioctls` for setting the text/graphics modes for VGA and AT&T enhanced CGA/EGA are different than in Release V.3.2. Any graphics application using these modes must be recompiled. If the application uses only standard EGA or CGA, then no recompilation is required.

Additionally, the KDVDCTYPE `ioctl` value has been changed. Any application dependent on this `ioctl` should be recompiled. (See EVGA mode changes in Chapter 5, "System Administration and Maintenance").

2.3.10 Mouse Driver Support of Multiple Terminal Assignments

A compilation problem prevents the mouse driver from supporting a configuration where one mouse is mapped to the console and another to a second console-like device. If a mouse must be assigned to the second console device, deconfigure the mouse from the console.

2.3.11 ioctl Command Conflicts

There are conflicts between `ioctl` command values in `kd.h`, `stermio.h`, and `termiox.h`.

The commands `SPECIAL_IOPRIVL`, `STSET`, and `TCGETX` all have the same value. This does not cause a problem because there are no drivers that recognize `STSET` and `TCGETX`, and support for `SPECIAL_IOPRIVL` is not yet implemented. Once support is added for `SPECIAL_IOPRIVL` and/or drivers are added that recognize `STSET` or `TCGETX`, these commands will not be handled correctly.

There are also potential conflicts between `STTHROW` and `TCSETX`, `STWLINE` and `TCSETXW`, and `STTSV` and `TCSETXF`. The `ioctl` command values in `stermio.h` and `termiox.h` will be changed in the future to eliminate these conflicts.

2.3.12 Cannot Cross Layers in shl

Do not attempt to display the output of a layer in another layer in shell layers when `-loblk` is off.

2.3.13 Scripts and Relative Symbolic Links

When a script is executed implicitly (e.g. with a `#!` as the first line in the script), and the command to be executed is a relative symbolic link to a relative symbolic link to the real file, then the script will not be found. The work around is to reference absolute symbolic links or hard links in scripts.

When a shell script is executed implicitly, with a `#!` as the first line in the script via a symbolic link, it will set `$0` to the name of the actual script rather than the name of the symbolic link. This may cause problems with scripts that check to see how they are called to determine their behavior. Currently, there is no workaround for this.

2.3.14 dd to Raw DMA Device with > 16MB Fails

On systems containing more than 16MB of RAM, data transfers such as `dds` may fail. The `dd` command does not abort, but rather terminates normally leaving corrupt data files

2.3.15 /usr/ucb/mt Command

The `/usr/ucb/mt` command is included for reference only. Use `/usr/lib/tape/tapecntl` command instead.

2.3.16 Dump Slice Names

On SCSI systems, the node for the dump slice is not created even if explicitly requested at installation time. A suggested workaround is to create the `/dev/dump` node with the `mknod` command and link `/dev/dsk/0s6` to `/dev/dump`.

2.3.17 Blank Interface Drivers

The implementation of `itoemajor` on the 386 is incorrect. In order to correctly use this implementation you must first find all external major numbers mapped to a given internal major number until the function returns `-1`. Currently, the 386 version returns the major number passed to it, since the internal and external major numbers are mapped one-to-one. Thus, a driver will never see a return of `-1` and will loop forever calling `itoemajor`, and deadlock your system. Therefore, we recommend that you do not use blank interfaces in your algorithms at this time.

2.3.18 format Utility and O_EXCL Flag

The format utility does not open the device to be formatted with the `O_EXCL` flag.

2.3.19 Window in Sockets Module

There is a window in the sockets module, `/boot/SOCKMOD`, in which it can do a `putnext` after the queue structure has been torn down. This may cause system problems.

2.3.20 Functions Removed from DDI Interface

Three functions were removed from the SVR4 DDI interface; `major`, `minor`, and `makedev`. The correct functions to use in their place are `getmajor`, `getminor` and `makedevice`, respectively. The removed functions are not documented and are not included in the final interface. The arguments for the removed functions and the remaining ones are the same.

2.4 SCSI DEVICE NOTES

2.4.1 LED on front of SCSI tape drive stays on

The light on the front of the SCSI Archive Viper tape drive stays on even after the tape cartridge has been rewound and removed.

To get the light to turn off, before removing the tape cartridge, type **`tapecntl -u`**. The tape will rewind and unload, then the light will turn off.

2.4.2 SCSI device driver CDROM Support

The current driver supports only Toshiba Model XM-3201B CDROM.

2.4.3 SCSI Configuration Considerations

When configuring your SCSI system, note the following:

- Only one Adaptec controller card can be used in the same system.
- Up to three Western Digital controller cards can be configured.
- Controller types (SCSI and/or non-SCSI) cannot be mixed. For example, don't use an ESDI or ST506 (non-SCSI) controller and a WD7000 or Adaptec controller in the same system.
- Don't combine a WD7000 and Adaptec controller in the same system.

2.4.4 Configuring Multiple WD 7000 Controller Cards

To configure the system for two WD 7000 (Western Digital) FASST SCSI Host Adapter cards, do the following:

1. Use the default configuration for the first WD7000 (IRQ 15, DACK/DRQ 6, BIOS ROM address 0xCE000, I/O address range 350-353).
2. Configure the second WD7000 with IRQ 11, DACK/DRQ 5, BIOS ROM address 0xC8000, I/O address range 330-333.
3. Configure the jumpers on the second WD7000 as follows:
   - W6 in and W9 in (to disable the floppy hardware)
   - W1: out out out out out (for IRQ Channel)
   - W2: out out in out out in out out out in out (for IRQ 11)
   - W3: in out out in in (for I/O address range 0x330-0x333)
   - W4: in in out in (for BIOS ROM address 0xC8000, which is not used).

   — NOTE —

   See the hardware reference manual that came with the hardware for details on how to set jumpers.
4. Disable the BIOS on the second WD7000 by doing one of the following:
   - If jumper W98 (near Z2, lower terminator) exists, remove it. (Newer WD7000 cards have this jumper.)
   - If there is no W98 jumper (and no CR5 LED and no F1 fuse), you have an older card and you must remove the BIOS ROM (U60) instead.
5. Install both WD7000s in the 16_bit expansion slots.
6. Connect a SCSI cable to each WD7000.
7. Connect the other SCSI devices.
8. Make sure that each SCSI cable is properly terminated at the ends.
9. Turn on power to all the hardware attached to the system.
10. Reboot the system.
11. Edit the `/etc/conf/sdevice.d/scsi` file so that it looks like the following:

```
scsi    Y    1    5    1    15    350    353    0    0
scsi    Y    1    5    1    11    330    333    0    0
```

12. Rebuild the UNIX kernel by typing `/etc/conf/bin/idbuild`.

You may need to remake the device nodes, depending on the devices hooked up to the two SCSI buses. For example, if each SCSI bus has a WD7000 (ID 7), a Winchester Drive (ID 0), and a tape (ID 3), then:

- `/dev/dsk/c0t0d0s0` is the system disk on BUS-0
- `/dev/rmt/c0s0` is the tape drive on BUS-0
- `/dev/dsk/c0t1d0s0` or 1s0 is the disk on BUS-1
- `/dev/rmt/c0s1` is the tape drive on BUS-1

To remake the device nodes, do the following:

1. Edit the appropriate file(s) in the */etc/conf/node.d* directory.
2. Change the line that reads `dsk/c0t1d0s0` (or 451s0) to `dsk/c1t0d0s0`.
3. Change the line that reads `rmt/c0s1` to read `rmt/c1s0`.
4. If you have three disks or three tapes in a system, you need to make the nodes for the third devices yourself. The minor numbers for the (n+1)th device are always `n + 16`.

2.4.5 SCSI disks need to support REASSIGN_BLOCKS command

SCSI disks that support SCSI CCS (Common Command Subset) don't necessarily support the REASSIGN_BLOCKS command. The SCSI disk driver (sd01) depends on the REASSIGN_BLOCKS command for dynamic bad block (or defect) handling.

Make sure your SCSI disk supports this command.

2.4.6 Optical Disks and SCSI Controllers

Optical disks connected to the SCSI port must meet the following requirements:

- 512 bytes block size
- SCSI device type= 0

Two devices (Maxtor-Tahiti-I and Sony SMO-E501) were tested, and several problems occurred:

- If an optical disk cartridge is not in the drive when UNIX boots, you get some SCSI error messages. You can ignore these messages.
- The drivers do not lock removable media into the drives, so you can, but should not, remove a disk with mounted file systems on it.

2.4.7 Maxtor XT-4000S SCSI Disks May Lock Up

SCSI systems with Maxtor XT-4170S or XT-4380S disk drives may hang due to disk drive lock-ups. This problem is indicated by the message:

```
SDI return code= 0xD0000009 (Job time-out).
```

Systems with a WD7000 controller recover by resetting the SCSI Bus. Systems with an AHA-1542 controller will sometimes recover by resetting the locked-up disk, and will sometimes hang.

If the read-ahead cache on the disk drive is disabled, the disk does not lock up.

2.4.8 Cannot boot from Third Partition on the WD7000 SCSI Controller

If your system has a SCSI WD7000 controller and the hard disk is partitioned with two DOS partitions and one UNIX partition, or one DOS partition and two UNIX partitions, if you try to boot UNIX from the third partition, the system may hang.

CHAPTER 3. FILE SYSTEMS

3.1 Virtual File System

3.1.1 Source Compatibility

VFS constitutes an extensive rewrite of the file system code. No substantial source compatibility should be expected for file system code written prior to SVR4.

3.1.2 Future Directions

Aspects of the VFS interface (particularly the interaction between file system code and the Virtual Memory system) are subject to change in future releases of the system. Absolute compatibility is not guaranteed.

3.1.3 File System Files

The information that was contained in `/etc/checklist` and `/etc/fstab` is now found in `/etc/vfstab`. The commands `fsck` and `ncheck` now use `/etc/vfstab` instead of `/etc/checklist`, and the `mount` command now uses `/etc/vfstab` instead of `/etc/fstab`. (See the `mnttab(5)` manual page). The file is now readable, and contains five fields.

3.1.4 Switchout

The following commands were coded to work on the VFS switchout mechanism: `clri`, `dcopy`, `df`, `ff`, `ncheck`, `fsck`, `fsdb`, `labelit`, `mkfs`, `mount`, `umount`, and `volcopy`. These commands have a generic module and file system specific modules which the generic will call. Commands operating on unmounted file systems shall either have the file system type supplied on the command line with the `-F` option, or an appropriate entry shall be provided by the administrator in `/etc/vfstab`.

3.2 File System Commands

3.2.1 clri Command

The `clri` command is currently working under the switchout mechanism but will be dropped in a future release. `clri`'s functionality is being replaced by `fsdb -z`.

3.2.2 fsstat Command

The function previously supplied by the `fsstat` command is now provided by `fsck -m`. `fsstat` is provided in SVR4, but its support will be dropped in a future release.

3.2.3 fsck Command

For the `fsck` command, matching in `/etc/vfstab` is performed on the `fsckdev` entry in each line, not the `special` entry. This implies that `mount /usr` works, but `fsck /usr` does not. In fact, `fsck <fsckdev entry for /usr>`, works.

The `mountall` code has been changed to use `fsck -m` to determine if a file system is suitable for mounting.

3.2.4 Future Directions

The format of the `mount` command will be replaced in a future release by a new format currently provided under the `-v` option.

Since the `ff` command is more flexible, it will eventually supersede `ncheck`. In a future release, `ff` will subsume all functionality of `ncheck` and `ncheck` will be removed from the system.

3.3 File System Types

3.3.1 s5

The s5 specific `fsck` command code has changed for size checking. The "possible file size error" that was suppressed with the `-q` option no longer exists. If the number of blocks in an inode exceeds the file size, the user is asked if the excess blocks should be recovered or deleted. If a directory size is not a multiple of the directory structure size, the user is asked if the size should be set to the address of the last byte of the file or left unchanged.

The option `-r` of the s5 specific `mount` command is provided in SVR4 for compatibility, but the function of this option is executed by the `-o ro` option. The `-r` option will be removed in a future release.

3.3.2 RFS

See Chapter 4, *Networking Notes*, for information on Remote File Sharing(RFS) implementation issues.

3.3.3 proc File System

Utilities that use `/proc` may fail if they try to read information about an existing hung process. They may sleep waiting to lock a process' information. If a particular process is hung, the utility may sleep forever. This problem is apparent if a user executes a utility that does locking in `/proc` (such as `ps` or `priocntl`), and the system currently has a live process that is hung after locking its process information.

3.3.4 UFS

The UFS file system is compatible with the BSD file system.

3.3.4.1 New Commands and Source Compatibility. A new command, `ff`, similar to its counterparts for the `s5` file system, is supported for UFS. This command is not contained in SunOS or BSD.

3.3.4.2 mount Command. The BSD "`mount -a`" option is not supported in UFS. `mountall` should be used to mount all file systems (including UFS file systems) specified in `/etc/vfstab`. The following options to `mount`, which are supported in SunOS, are not supported for UFS in SVR4: `grpid`, `quota`, `noquota`, `noauto`. The "`suid`" option is not supported; UFS file systems are mounted with `setuid` execution allowed by default. The "`nosvid`" option is available to mount UFS file systems with `setuid` execution disallowed.

3.3.4.3 fsck Command. The `-p` option to `fsck` (supported as suboption `p` under `-o`) does not work in parallel, as it does in BSD/SunOS.

Unlike SunOS/BSD which uses the errno message `EDQUOTA`, no special `errno` is returned when a user tries to exceed quota limits. Instead, an `errno` is returned that indicates failure due to the lack of space. The `errno` message `ENOSPC` is the most common.

3.3.4.4 ufsdump/ufsrestore. `ufsdump` sometimes reports more blocks written than it actually writes. Therefore, the blocks reported by `ufsrestore` and `ufsdump` may be slightly different. The dump is executed correctly and is usable. If there is a discrepancy between the two commands on the number of blocks in the dump, the number reported by `ufsrestore`, which should be slightly less than that reported by `ufsdump`, is correct.

3.3.4.5 mkfs and UFS. `mkfs` on a UFS file system has a compatibility mode that lets you limit the number of inodes to 64K. This provides compatibility with System V Release 3.2 applications that require access to inode numbers. This compatibility mode is used by default in `disksetup` or by invoking `mkfs` with the `-C` option.

3.3.4.6 MAXFRAG. This release of System V Release 4 redefines `MAXFRAG` to 8. Any file systems created on a previous version using a different `MAXFRAG` should be mounted read-only.

3.3.4.7 proto file. Neither the BSD `mkproto` command nor the s5 `mkfs -p` option is supported in UFS.

3.3.4.8 ncheck Command. The `-m` option works only for inodes specified under the `-i` option.

3.3.4.9 quota Command. The `quota` command only works correctly when all UFS file systems that are currently mounted have a quota file. An empty file named `quota` should be placed at the root of each UFS file system.

3.4 Symbolic Links-Compatibility SVR4/ SunOS

The following list references a few of the incompatibilities between SVR4 and SunOS features.

3.4.1 link System Call

SVR4 allows hard links to symbolic links; SunOS does not.

3.4.2 chown System Call

SVR4 `chown` follows symbolic links and changes the owner and group of the referenced file; SunOS `chown` changes the owner and group of the symbolic link.

3.4.3 lchown Command

This command is new in SVR4. It is the same as `chown` except `lchown` does not follow a symbolic link. It changes the owner and group of the symbolic link.

3.4.4 file Command

By default, in SVR4 the `file` command follows symbolic links. The `-h` option says to not follow symbolic links; the default for the `file` command in SunOS is to not follow symbolic links, and to use the `-L` option to follow them.

3.4.5 tar Command

SVR4 uses the `-L` option to follow symbolic links; SunOS uses the `-h` option.

3.4.6 find Command

SVR4 uses the `-follow` option to follow symbolic links; SunOS does not.

3.4.7 cpio Command

SVR4 has `-L` option to follow; SunOS does not.

3.5 /etc/ vfstab Entries Created by diskadd

If the `diskadd` command is run more than once, the entries in `/etc/vfstab` for that drive may not be deleted. If you intend to run the `diskadd` command more than once for a given drive, delete the entries for it in `/etc/vfstab` prior to running `diskadd`.

3.5.1 Diskadd and mount points

The `diskadd` operation allows the user to re-use mount points that already exist in the `/etc/vfstab` file. When re-used mount points are mounted, the `diskadd` operation fails when attempting to re-mount the selected mount point. When re-used mount points are not mounted, the `diskadd` operation succeeds.

CHAPTER 4. NETWORKING

4.1 TCP/IP (DARPA Internet Protocol Suite)

4.1.1 Pseudo-terminals

Pseudo-terminals must be configured for TCP/IP services, such as `telnet` and `rlogin`, to work. These may be selected when the NSU package is installed.

4.2 NFS

4.2.1 Use NFS bg in /etc/vfstab

NFS file systems specified in `/etc/vfstab` should use the `bg` option to prevent the system from hanging or looping at boot time when the remote system is down.

4.2.2 Unsharing Advertised Resources

The `unshareall` command may not always unshare an advertised resource. This problem is specific to the NFS `unshare` command.

4.2.3 Automounter

Direct map automounters do not always clean up after themselves. They may leave files that are symbolic links to automount daemon mount points for which the automount daemon no longer exists. Attempts to access these files will result in `server not responding` messages. These files can only be removed by unlinking them, then running `fsck`, or by rebooting and then removing any remaining links. The only other workaround is to avoid using direct maps with the automounter.

Note that the automount daemon for direct maps mounts itself on the mountpoint specified in the map, while the automounter for an indirect map mounts itself on the directory specified as the root for the indirect mount, i.e., one level above the mount point for the resource in the indirect map. This can be misleading to the user because the user may not expect anything to be mounted above the mount point of the resource. Because the daemon mounts itself there, the previous contents of the directory are covered for the duration of the life of the automounter. As with all file system mount points, it is a good policy to use empty directories as automount root mount points.

There is no entry in the `/etc/mnttab` file corresponding to the mount of the automount daemon on the automount root mount point. Therefore, it is possible for a user attempting to mount a file system on a given mount point to get a `mount point busy` message even though that directory is not noted in the `/etc/mnttab` file as an existing mount point.

4.2.4 Lock Manager

The lock manager does not always detect deadlock.

4.2.5 Loopback

If a large read or write operation to a loopback mount with `biod`'s running is interrupted, some of the `nfsd`'s can hang while waiting for the STREAMS code to free a STREAMS message block.

4.2.6 Secure NFS mount

If a secure NFS mount is interrupted while waiting for the keyserver to respond, it is possible for the file system to be mounted, but without a corresponding entry put in the `/etc/mnttab` file. This occurs because the signal will not be handled until after the mount system call has succeeded and returned to the `mount` command. The command process will then be killed before it writes to `/etc/mnttab`.

4.2.7 Transport Independence

NFS can run over multiple transports simultaneously, provided that:

a. both the client and the server agree on the maximum packet size for a given transport; or, the `-o rsize=,wsize=` options are given to the `mount` command to reduce the read and write sizes to a value small enough for both machines (the minimum of the packet sizes of the two machines' transports). Note that the read and write transfer sizes should be about 430 bytes smaller than the transport's packet size to allow for RPC headers.

b. the transport can accept packets of at least a minimum size, of approximately 1400 bytes.

The NFS daemon (`nfsd`) does not listen on a specified address over transports other than UDP. Therefore, if the `nfsd`'s are killed and restarted, the new `nfsd` may listen on an entirely different address causing any previously existing mounts to fail.

If the NETPATH environment variable is set, then at least one of the transports in the NETPATH path should be connectionless.

4.2.8 Compatibility with SVR3.2

A NFS mount from SVR4 to a SVR3.2 system running Lachman NFS 3.2.4 does not work. If the repeating error message, `Authenticate error`, is displayed on a SVR4 system, `root` on that system belongs to too many groups. To mount a Lachman 3.2.4 system, `root` must not belong to more than eight groups.

The workaround is to edit the `/etc/groups` file. The line for root should only contain seven entries (including `root`). Then log out and back in.

— NOTE —

All users belong to group `other`, even though it isn't specified in their line in the `/etc/groups` file.

4.3 RFS

Remote File Sharing (RFS) has been implemented as a file system type under VFS. Even though the new implementation fully supports the pre-SVR4 protocol, the new protocol has implications for compatibility and inter-operability with previous versions of RFS.

Using `df` with either the `-n` or `-g` options on a remote resource advertised from a pre-SVR4 system will give `unknown` for the fstype.

The `-c` option to the `rfs` specific `mount` command function is done by `-o` nocaching.

The `-d` option for the `mount` command is provided for compatibility, but its function is replaced by `-F rfs`. So, the `-d` will be removed in a future release.

4.3.1 Known Problems

Facilities new to SVR4 cannot be provided by an older RFS server. Although a SVR4 RFS client can create a dynamic shared library on a pre-SVR4 server, the shared library cannot be `mmap`ped from that server, because the pre-SVR4 protocol does not support file mapping or paging. The `rename` system call (used by the `mv` command) is not fully supported between SVR4 clients and SVR3 servers. An attempt to `rename` a directory on an SVR3 server will fail with the error `EISDIR`.

RFS does not support `mmap`ping of character devices or the allocation of remote swap files.

Users and administrators of RFS clients and servers are reminded that the interpretation of absolute symbolic links encountered on the server can lead to unexpected results because they are relative to the root directory of the client.

Setting up RFS via sysadm menus is not currently possible. The menu function for RFS is unpredictable and problematic.

4.3.2 Swap Control

Swap will not work over RFS (i.e., a swap device or file cannot be created on a resource mounted via RFS). If this is attempted, an `ENOSYS` error will result.

4.4 RFS: sysadm Initial Remote File Sharing Setup

When using the `add_nameserver` (Adds Domain Name Servers) feature, there are two errors that you may encounter. The first possible error message is:

```
Name to Address Mapping has not been set up correctly.
Select the Name to Address Mapping Management task from
the remote_files menu to setup Name to Address mapping.
```

If you are using TCP/IP as the transport provider, this may indicate that there is not a listen service in the `/etc/services` file. When using the `Name to Address Mapping` to add the entry, the service will be listen and the port will be the port value used to setup the listener. The default listener port value for `tcp` is ACE in hexadecimal (which is 2766 in decimal). The `Name to Address Mapping` menu expects the value to be in decimal. To get the listener address, you could type **`nlsadmin -l tcp`**. The port number will be the 5th through 8th numbers.

— NOTE —

RFS should be set up using command line input versus OA&M menus.

4.5 Secure RPC

4.5.1 Secure RPC with RFS

The Secure RPC administrative files `/etc/masters` and `/etc/slaves` contain the unames of RPC masters and slaves respectively as documented in *Programmer's Guide: Networking Interfaces*. However, if RFS is used to share the files from the masters or the slaves, then those entries in either file should contain the RFS domain name for that master/server followed by a dot (.) and the uname, i.e., `rfsdomain.uname`.

One of the Secure RPC administrative tasks of a slave server is to share its `/etc` directory, writable to its master and readable to its clients.

When NFS is used to share the `/etc` directory, the `share` command should be in the form `share -F nfs -r rw=p,root=p,ro=i:j:k /etc`, where `p` is the slave's master and `i,j,` and `k` are the slave's clients.

4.6 OA& M - Remove System Name function

Due to some data parsing problems, the OA&M menu item for removing a system name from a basic networking database does not function. This item was intended to be an interface to the `/etc/uucp/Systems` file. Administrators who want to remove a system from the list of systems that the machine communicates with must edit and delete the appropriate system name line in the `/etc/uucp/Systems` file.

4.7 Cable Disconnections During I/O via tty ports

If a cable is disconnected while reading from or writing to an asynchronous communications port (COM1 or COM2), the process will get a `0` in the case of reading (for end of file) and an I/O error in the case of writing.

When the cable is reconnected to the port, the user will continue to get the above return values from read or write, rather than the input or output starting back up again successfully. The only way around this is to close and then reopen the device.

An explanation for this behavior is that upon detecting "carrier loss" (disconnecting the cable), the driver sends the message `M_HANGUP` to the stream head. When the stream head recognizes this, it sets a flag such that from then on, any further reads or writes will return an I/O error or `0`, respectively, to the user. When the carrier comes back (by connecting the cable), the flag is not reset since the streamhead does not know anything about it.

4.8 libnsl and the ABI

In previous releases, three routines required for `libnsl` by the ABI were actually placed in a different library. In this release, `getpublickey` and `getsecretkey` have been moved to match the ABI. Another routine, `svc_getargs`, will be dropped from the `libnsl` requirement in subsequent issues of the ABI.

CHAPTER 5. SYSTEM ADMINISTRATION/ MAINTENANCE

5.1 Improved Backup and Restore Operations

5.1.1 Extended Backup and Restore

If the user would like to use the extended `backup` and `restore` capabilities provided by OA&M, the following changes must be made,

For the `backup` section, edit `/usr/sadm/sysadm/menu/backup_service/ext.menu` and add the line:

```
extended^Extended Backup Services^backup.menu
```

For the `restore` section, edit `/usr/sadm/sysadm/menu/restores/ext.menu` and add the line:

```
extended^Extended Restore Services^restore.menu
```

— NOTE —

Use of the extended `backup` and `restore` capabilities requires an advanced level of familiarity with SVR4's `backup` and `restore` commands. (For more information, see the *System Administrator's Guide*).

5.1.2 Source Compatibility

System administrators must recreate the online backup schedule using the `bkreg` command.

5.1.3 Future Directions

The `ckbupscd` command will not be supported in future releases.

5.2 Specific Backup/ Restore Notes and Workarounds

5.2.1 backup and cpio

Do not use `cpio`, `volcopy`, or `tar` to backup `/usr`. Only use the `backup` command to backup `/usr`.

5.2.2 -o Method Option

The `-o method` option overrides label checking on `backup`. The override allows a backup to a tape or disk that has a label different from that specified for the operation. Because `restore`, and `rsoper` always check labels, non-NULL labeled media should always be supplied in any backup operation, even with label checking override. The -o option should be avoided if possible.

5.2.3 SCSI Tape Driver and Supported Block Sizes

The SCSI st01 tape driver does not support actual block sizes greater than 60 KB on the tape. The size of the requested I/O from the application should be an integral multiple of the actual block sizes of the blocks on the tape. For example, if data was written to a tape with 5K block length the tape should be read with 5K, 10K, or 15K reads.

5.2.4 Backup and Device Names

When using the `backup/restore` commands, a specific device name must be specified not just the device group. If for some reason, the device group must also be specified, the "spool" entry in `/etc/device.tab` must be moved to the end of the file.

— NOTE —

Save a copy of the `/etc/device.tab` file as it is recreated each time the system boots.

5.2.5 Fimage Archives

The `fimage` method restores an entire file system. If such an archive is online, and the file system is unmounted, the restore will occur automatically. A slice may be unintentionally overwritten if the administrator requests a `restore` and such an archive is available.

5.2.6 Backup Strategy

Both `fimage` and `ffile` methods are complete file system backup strategies and are mutually exclusive. Correct strategy is to use either `fimage` or `ffile` in conjunction with `incfile`.

5.2.7 Viewing Archive Labels

The command `rsoper -n -d` allows an administrator to view the label of the archive on the device specified. This is useful if external labels are damaged or a backup history is lost.

5.3 /etc/ bkup/ rsstatus.tab: Not Created During Installation

The file `/etc/bkup/rsstatus.tab` is not created when the system is installed. If you attempt to query this file using the Extended Restore function under OA&M, you will get a message to this effect. This is not serious as the file will be created when the first Extended Restore is executed under OA&M.

5.4 Console Logging

5.4.1 Future Directions

The `/dev/console` and `/dev/syscon` special files are still available for use in this release. Access to a console by `/dev/console` may be unavailable in future releases.

5.5 System Administration Menus sysadm

A new screen-based administrative interface, `sysadm`, is introduced in SVR4. This interface uses FMLI utilities and can be operated with the function keys on most terminals.

5.5.0.1 In a limited number of cases, users may find that keyboard function keys may need to be reset after the termination of a `sysadm` session.

5.5.1 Future Directions

See your keyboard or terminal manual for more information. The administrative logins (i.e. `setup`, `powerdown`, `makefsys`, `mountfsys`, `umountfsys`, and `checkfsys`) no longer have entries in `/etc/passwd` and may not be supported in the next release.

5.6 Device Management

A table driven device management facility is provided in this release. The administration feature set depends upon the information in a device table to access tape drives, floppy disks, and hard disks for software installation, backup/restore operations, and device access throughout the `sysadm` interface. The table may be populated using the `storage_devices/descriptions/add` task on the `sysadm` interface.

5.6.1 Source Compatibility

The SVR4 software installation feature recognizes packages in the pre-SVR4 format and cannot be used to install and remove such packages. The SVR3.2 commands `installpkg`, `removepkg`, and `displaypkg` are provided for compatibility. It is recommended that new software packages be created and existing packages be converted, where feasible, using the new format in order to take advantage of package auditing capabilities, and future enhancements. Add-on packages that supply `FACE` interface tasks should use the new `sysadm` hierarchy.

5.7 User and Password Administration

5.7.1 Future Directions

The `pwconv` command will not be updated in future releases to work with new identification and authorization database files.

The `passmgmt` command will be removed from distribution in the next release. The functionality has been replaced by the SVR4 `useradd`, `usermod`, and `userdel` commands.

5.8 EVGA Mode Changes

For the video cards that are described on the `evgainit` manual page, there are new mode switching constants in `kd.h`. They are all prefixed with `SW_GEN_` followed by the resolution. These should be used to set any of these video cards to graphics modes with resolutions higher than 640x480. For example, to set the video card to 800x600, the `SW_GEN_800x600` constant would be used. The ioctl to set the mode will succeed if the video card in question supports the specified resolution, otherwise it will fail and set errno to `ENXIO`. In order to set the video card mode, first open `/dev/video` and then do an `ioctl` on the file descriptor with the appropriate mode switching constant. For example:

```
fd = open("/dev/video", O_RDWR);

ioctl(fd, SW_GEN_800x600, 0);
```

5.8.1 Supported Video cards

The following video cards are supported in the 800x600 pixels (16 color) VGA mode:

— AT&T VDC 600

— ATI VGA Wonder

— Video7 VRAM VGA

— Video7 Fastwrite VGA

— Sigma VGA/H

— Dell VGA

— Paradise 16PLUS

— Techmar VGA

The following video cards are supported in the 1024x768 pixels (16 color) VGA mode:

— Paradise 1024

— Orchid Pro Designer VGA

— STB VGA Extra/EM 16

5.9 Commands

5.9.1 pkgadd

There is a problem with the OA&M `pkgadd` if the files contained in the OA&M package have group ownership values not contained in the system's `/etc/group` file. Enter the group owner ID values in the `/etc/group` file.

Labels must be specified for all backups and there must be enough for the number of volumes to be created. The option to overwrite the label `-o`, should not be used.

5.9.2 pkgmk

When using `pkgmk` to create a file, the capacity specified should be slightly less than the true capacity of the device. The simpliest way to do this is to add `-l <sm. size>` to the `pkgmk` command line, where `<sm. size>` is roughly equal to the size of the `pkginfo` file (in 512 byte blocks) plus 1. If the new value is still too large, a warning about no space being left on the device will be sent to the console.

5.9.3 pkgrm

If a package removal fails, try to remove the package again. If the removal fails again, then try to remove the lock file by doing the following:

1. Type **`cd /var/sadm/pkg`**
2. Type `cd` < *pkgname*> where < *pkgname*> is the directory corresponding to the package you are trying to install. Check the "NAME" field in the `pkginfo` file to assure the correct directory has been selected
3. Remove the file called `!R-Lock!`.

5.9.4 pwck

The command `pwck` only lists information about `/etc/passwd`, and not `/etc/shadow`.

5.10 OA&M Screens and Help

The following menus have no Help text:

- restore_services/extended/status/full
- backup_services/extended/status/full
- applications
- network_services

Consult Networking documentation for help.

5.11 Setting up RFS through sysadm

Setting up RFS via `sysadm` menus is not currently possible. The menu function for RFS is unpredictable and problematic.

5.12 Terminal Cable Recommendations

Do not use a terminal cable that has DCD and DTR looped back to each other. If you must use a loopback cable, do not attach the cable from the terminal to the COMM1 or COMM2 port until after you have started up a login on the port, or you will get a carrier detect error or the system will hang.

5.12.1 /etc/ttytype File Format Incorrect for Non-ansi Terminals

`ttymon` reads the `/etc/ttytype` file to determine the terminal type for the login session on that port. ansi is the default terminal type. For example, the default entry for an ansi terminal in the `/etc/ttytype` file looks like this:

```
ansi        tty00
```

If you are using a non-ansi terminal (such as vt100), change the line to look like this:

```
vt100     00
```

Now, when you log in at port `tty00`, the default terminal type will be set to `vt100`.

CHAPTER 6. REAL TIME PROCESSING

6.1 High-Resolution Timers

6.1.1 Source Compatibility

In SVR4, `gettimeofday` and `settimeofday` take one argument. The second argument used with the respective BSD calls (and in the BSD compatibility library) has long been obsolete and is ignored.

In `libucb.a`, the `alarm` and `sleep` functions are written using `setitimer` [see `getitimer`(3C)]. In `libc.a`, `alarm` is independent of `setitimer`. In SVR4, a `sleep` following a `setitimer` wipes out knowledge of the user signal handler. Therefore do not use `setitimer` with `sleep`.

6.1.2 Future Directions

These routines will be included in SVR4 until the POSIX P1003.4 standard on Real Time System Interfaces is completed.

CHAPTER 7. C PROGRAMMING LANGUAGE

7.1 Directory Layout

In SVR4, the directory layout of the compilation system changed. While header files are still located in `/usr/include`, all software development tools that were installed in `/bin` or `/usr/bin` in previous releases are now installed in `/usr/ccs/bin`. Similarly, all software development tools, libraries, and other files that used to be installed in `/lib` or `/usr/lib` are now installed in `/usr/ccs/lib`.

Object files created by the SVR4 compilation system are in a new format called ELF (Executable and Linking Format). While the compilation system tools support linking with and analysis of object files in the older format (COFF), there may be users who have older C compilation systems they wish to continue to use or who have non-C compilation systems that depend on being able to generate COFF-style object files. We wanted to allow older compilation systems to coexist with the SVR4 system, but we have no control over how older systems get installed and we cannot predict what directory dependencies they might have. Therefore, we chose to move the SVR4 compilation system into new directories, thus stepping out of the way of those older systems.

The installation process for the SVR4 compilation system will automatically add the directory `/usr/ccs/bin` to the default command search path of ordinary users [see `sh`(1)]. This should assure that most compilation command scripts or `make` scripts will not have to change to allow for the new directory structure and allow the directory structure to be transparent.

To ensure compatibility for those scripts that access compilation system commands or `/lib/cpp` by absolute pathnames, symbolic links for all commands in `/usr/ccs/bin` and for `/usr/ccs/lib/cpp` will automatically be made at installation time if no older compilation system is already resident on the system. The links will be made to files of the same names in `/usr/bin` and `/usr/lib` respectively. These links are intended as a transition aid and will be removed in a future release.

7.2 External Data/ Automatics Order

As always, the order of external data and automatics on the stack should not be relied on. In particular, the compilation system is free to do register allocation.

7.3 long double

The `long double` data type is not presently supported as stated in the ABI.

Although 12 bytes are allocated for a long double currently, only eight are used. When a long double is passed as an argument, only eight bytes are passed, and conversely, any function receiving a `long double` must expect the stack frame to look as if a double

were passed.

As a result, library functions that are ABI conformant and expect a `long double` argument or that return a `long double` will not be callable from user code.

The `scanf()` and `printf()` functions treat `long double` specifiers as if they were double specifiers, and `long double` arguments to `printf()` are treated as double.

Full `long double` support will be available in the future.

7.4 Floating Point Arithmetic

All floating point operations are performed in double extended precision, as permitted by the ANSI C standard, until explicitly stored in memory or until explicitly casted to a type other than `long double`. The following example illustrates the problem; it will print `not equal`.

```
#include <math.h>

main()
{
 double d = (nextafter(1.0, 1.1) + 1.0) / 2.0;

 if (d != ((nextafter(1.0, 1.1) + 1.0) / 2.0))
   printf("not equal\n");/* expression is between [1,d] */
 else      printf("equal\n");
}
```

7.5 ifiles No Longer Supported

The `ifiles` feature of the link editor command language is no longer supported. You should switch to the new `mapfiles` feature, which subsumes much of the functionality.

7.6 Null Pointer References

Some null pointer references in the source code have not been fixed.

7.7 Optimizer and asms

The optimizer does not handle `asms` that change the depth of the stack.

7.8 Performance Tradeoffs

The initialized arrays `_ _iob` and `_ _ctype` are defined in the dynamic C library since they are referenced by many of the library functions. They may also be referenced directly from the user's code through macros such as `getchar` and `isdigit`. Since the user's code is not typically compiled as position independent, space for these symbols must be allocated in the `a.out`'s data segment. At process startup time, the dynamic

linker changes the global offset table entries in the library to point to the `a.out`'s symbols, if present, so that both the library and the `a.out` will reference the same object.

`__iob` and `__ctype` are fairly large arrays, and the method chosen to initialize the arrays in the `a.out`'s data segment can have an impact on overall system performance. In one method, both `libc.so` and `libc.so.1` are built with the files (`data.o` and `__ctype.o`) containing the initialized data. Executable files linked with `libc.so` will have the initialized arrays in their `.data` sections.

The method used in this product is to replace the definitions in the archive `libc.so` with uninitialized objects (`.bss` symbols), and add code to the dynamic linker to copy the initialized data from the shared library into the `a.out`'s `.bss` section at process startup. This alternative typically makes executable files smaller and trades off disk I/O for longer startup times.

7.9 Dynamic Libraries

Most of the executables in UNIX System V Release 4 Version 3 are compiled with dynamic libraries. To determine which dynamic library (if any) the executable uses, type:

```
ldd a.out
```

This will display the dynamic library being used with this executable. However, it nothing displays if the executable is compiled with archive libraries.

7.10 Shared Libraries

Existing static shared libraries will continue to work with this release. You cannot, however, create new ones, as the commands `mkshlib` and `chkshlib` are no longer supported. This functionality has been completely replaced with the new dynamic linking feature.

7.11 libPW relocation

The pre-SVR4 contents and new functionality of `libPW` are located in `libgen`.

7.12 Commands and Functions

7.12.1 ctrace Command

The `ctrace` command does not handle `asms`.

7.12.2 dump Command

The `dump` command returns a status code indicating successful completion (return code set to zero) despite the following cases of bad input:

- number out of range
- invalid range
- section not found
- no such file or directory
- invalid file type
- bad line info section

7.12.3 ld Command

Error messages from the link editor that refer to I/O errors may be caused by an inability to create a file. This may be caused by permission problems, file space limitation, or `ulimit` problems.

Error messages from the link editor that specify "`vm stats`" errors or "`output file space`" may result from running out of system swap space.

7.12.4 lint Command

`lint -p` gives the diagnostic, `pointer cast may be troublesome`, when two pointers differ only by a `const`. (For example, `const int *` versus `int *`.)

`lint`'s `printf` format checks do not recognize positional parameters. Therefore, `printf("%1$s", s);` will yield the warning "`too many arguments for format`."

7.12.5 lprof Command

`lprof` does not ignore C code included via a header file. When it encounters a function that is defined in a header file, it begins outputting line numbers and line counts. Since this code does not appear in the original C file, the line counts will be off. A workaround is to place the C functions from the header file(s) into `.c` files and compile them as separate modules.

`lprof` will fail if the executable being profiled is built using two or more object files with the same basename, e.g., `directory1/fun.o` and `directory2/fun.o`. It will not be able to read the `.cnt` file created when the program is executed.

7.12.6 nm Command

The `nm` command returns a status code indicating successful completion (return code set to zero) when it encounters and warns about a truncated file.

7.12.7 SCCS Commands

Using the `delta` command on a text file that contains a line whose line length exceeds `BUFSIZ` (set to 1024 as defined in `stdio.h`) may cause an infinite loop or core dump.

7.12.8 sdb Command

Assembler routines that use the frame pointer and argument pointer registers in non-standard ways may cause erroneous stack traces in `sdb`.

Header files that include executable code may cause confusion with various tools (such as `sdb`) about line numbers. The result is that output from the tools relating to a function described in a header file may be associated with the wrong line number.

When a process is grabbed (via `sdb /proc/123`, for example), you may examine variables, instruction step, quit, continue, or kill the process. However, some breakpoints and statement steps will not work.

7.12.9 free Function

A `malloc`ed region of space can be freed only once. Freeing the same region of space more than once has undefined behavior.

7.12.10 mmap Function

Dynamically linked `a.out`s that depend on a shared library that resides on a file system that does not support the `mmap` function call (e.g., the BFS file system) will not execute.

7.12.11 nlist Function

The `nlist` function has been moved from the C library (`libc`) to the ELF library (`libelf`). This means the `make` lines of those programs using `nlist` need to be changed to explicitly look in `libelf`: `cc -o prog prog.c -lelf`

7.12.12 realloc Function

The `realloc` function has been changed for ANSI C conformance. If its size argument is zero, the object pointed to by the first argument is freed. The previous version of `realloc` simply returned `NULL` in this case.

7.13 Bitfields

Bitfields declared without a specific signedness will be treated zero-extended. The one exception is if the size of the bitfield matches the size of the type. For example, in the

following code, true will be printed:

```
struct {
            int bf: 4;
} x;

x.bf = 0xf;          /* fill bitfield with all ones */

if (x.bf < 0)
            printf("true0);
```

On the other hand, in the following code, true will not be printed:

```
struct {
            int bf: 32;
} x;

x.bf = 0xffffffff;  /* fill bitfield with all ones */

if (x.bf < 0)
            printf("true0);
```

In the future, the latter case will treat `x.bf` as an unsigned quantity, as per ANSI's value-preserving rules.

CHAPTER 8. LINE PRINTER SPOOLING UTILITIES

8.1 lp Package Enhancements

The SVR4 Line Printer (LP) Spooling Utilities package has been enhanced to include the following new features:

- Inclusion of new networking feature.
- Addition of new `lpsystem` command to define remote systems.
- Conversion of printer administration to the new OA&M System Administration Menu format plus inclusion of the following new functionality:
  - — Addition of class administration feature
  - — Addition of request administration feature
  - — Addition of priority administration feature
- Implementation of `cancel -u` option that provides ability to cancel all requests submitted by specified users.
- Ability to define a printer with multiple printer-types or content-types using the `lpadmin` command.
- Inclusion of PostScript filters.

Please refer to the manual pages and the *System Administrator's Guide* for detailed information regarding the new features listed above.

8.2 Compatibility

The SVR4 Line Printer Spooling Utilities is not source compatible with SVR3. Specifically, the new networking capability is based on new functionality provided in SVR4.

8.3 Known Problems

8.3.1 Networking

Although the new Line Printer Spooling Utilities networking feature works, it has not been thoroughly stress tested. It is possible that stressing the network capability may result in suspending LP network activity requiring the administrator to restart `lpsched`.

It is also possible that the networking feature may fault when attempting to network with a BSD system.

SVR4 does not support the `-R` option to the `lpstat` command, as documented on the `lpstat(1)` manual page. This option should not be used.

If a user's remote print request is canceled by the administrator on the remote machine, the user is not notified by mail. Canceling the request from the local machine generates the correct mail notification. In both cases the request is successfully canceled.

8.3.2 BSD Compatibility Commands

The BSD compatibility commands `lpc`, `lpq`, `lpr`, and `lprm` are operational, but may not be fully functional. `lprm` has only partial functionality. Please use the AT&T Line Printer Spooling commands instead.

8.3.3 Line Printer Spooling Utilities Directory Structure

In this release, the LP spooling directory structure has been changed to follow the new standard directory conventions; many files have been moved but the formats have not been changed. Please refer to the *System Administrator's Guide* for details on the new structure.

8.3.4 Line Printer Spooling Utilities over RFS

Due to directory structure changes, SVR4 LP is not compatible with SVR3 LP across RFS. Note that when you run the Line Printer Spooling Utilities over an RFS network, all machines in the network must be running the same version of LP.

8.4 lpstat Command

If the output of `lpstat -o -l` always shows that a particular `lp` request is canceled, then the following script should be run as root:

```
lpshut
cd /var/spool/lp
find requests tmp ! -type d ! -name .SEQF -exec rm {} ;
/usr/lib/lp/lpsched
```

8.5 Printer Interface Concerns

Since an Apple laser printer interface is not standard, a standard RS232 printer cable cannot be used. The cable used must tie `DTR` to `DCD` to configure printer with `O_NDELAY` flag on.

8.6 Batch Postscript

Many postscript printers support the option to run the printer in batch postscript mode over a parallel port. This is not supported by the line printer system. You must use interactive postscript mode over a serial line.

8.7 Non-ascii Printer Access Via the Network

If you want to access a non-ascii printer via the network, you must do the following:

1. On the system to which the printer is directly connected (the server), make sure the "Printer Type" and "Content Type" are defined as "PS" and not "postscript". To determine this, type **`lpstat -p <`** ***printer name*** **`> -l`**
2. As root, specify the name of the system which has permission to access the printer via the network using the `lpsystem`(1M) command.
3. On the client system, use the `-I` option of the `lpadmin`(1M) command to specify the content type of all files you anticipate sending to the printer. For example, if you anticipate sending troff, postscript, and ascii files to a printer called `qms` on a system called `inbox`, type:

   ```
   lpadmin -p qms -T PS -I postscript,troff,simple -s inbox!qms
   ```

4. You should now be able to print a job using the `lp`(1) command. You must specify the content type of the file you are sending. See the `lp`(1) command in the *User's Reference Manual* for more information.

CHAPTER 9. kdb MAN PAGE

9.1 INTRODUCTION

This chapter provides a copy of the `kdb(1)` man page which belongs in the *System V Release 4 Programmer's Reference Manual*.

NAME

kdb – kernel debugger

SYNOPSIS

kdb

DESCRIPTION

KDB is a kernel debugger that works like a Reverse Polish Notation (RPN) calculator. KDB can set breakpoints, display kernel stack traces and various kernel structures, and modify the contents of memory, I/O, and registers. The debugger supports basic arithmetic operations, conditional execution, variables, and macros. KDB does conversions from a kernel symbol name to its virtual address, from a virtual address to the value at that address, and from a virtual address to the name of the nearest kernel symbol. You have a choice of different numeric bases, address spaces, and operand sizes.

This is an advanced tool, only for those who are thoroughly familiar with the UNIX kernel. Because UNIX systems differ, you could possibly damage your system by following some of the examples in this discussion.

You can invoke the debugger by using the kdb command or the `sysi86(SI86TODEMON)` system call on all systems, <Ctrl-Alt-d> (from the console only) on an AT bus system, or the interrupt character (from the console only) on a Multibus system. In addition, KDB is entered automatically under various conditions, such as panics and breakpoint traps. Any time the kdb>> prompt appears, you are in the debugger. I/O is done via the console (kd), or a serial terminal.

To exit the debugger, type <Ctrl-d> or q.

When you exit and re-enter the debugger, its state is preserved, including the contents of the value stack.

USING KDB AS A CALCULATOR

KDB operates as an RPN calculator, similar to *dc*(1). This calculator has a 32-level value stack for storing results and intermediate values. Commands and values you enter operate on the value stack, which is an internal data structure in KDB. It has no connection with the kernel stack or any other stack in the system.

To use KDB, at the kdb>> prompt type one or more items (values or commands) on a line. Separate items with spaces or tabs. Press <Enter> to end a line and send its contents to KDB for processing. Each item is processed separately, from left to right.

The values can be:

Numbers. Use positive or negative integers. Numbers must begin with a digit, or a minus sign for negative numbers. Begin octal numbers with "0o" and hex numbers with "0x". Otherwise, numbers are assumed to be in the default base — the default is hex, unless you change it. (See "Resetting the Numeric Base" for instructions.)

Character constants. You can have KDB convert characters to a number by entering one to four characters inside single quotes. C-style escapes are supported in character constants.

Strings. Use C-style strings, enclosed in double quotes.

Kernel symbol names. When you type a kernel symbol name, its address is pushed onto the value stack.

When you enter a number or a string, it is pushed onto the value stack, becoming the new TOS (Top Of Stack). Values remain on the value stack until they are popped off as a result of a command.

In the descriptions below, [TOS] means the value on the top of the stack and [TOS-1] means the value just below it (pushed previously).

Stack Operations

KDB provides these commands for examining or changing the value stack:

`stk`	print all values on the stack
p	print [TOS]
dup	push [TOS]
pop	pop 1 value
`clrstk`	pop all values

- **stk** For example, starting with an empty value stack, this input:

```
5 "xyzzy" 7 stk
```

displays the entire stack:

```
5
"xyzzy"
7
```

- **p** At this point, the input:

```
p
```

displays the top value on the stack, which is:

```
7
```

The next example uses the p command to display the address of a kernel symbol. The input:

```
lbolt p
```

produces an address something like this:.

```
D01821BC
```

- **dup** This command is useful when you want to use a value twice in a calculation. For example:

```
5 3 * dup 2 + * p
```

would produce the output:

```
FF
```

which is the value of `(((5 * 3) + 2) * (5 * 3))`.

- **pop** This command removes the top value from the value stack. For example, if this is the stack:

```
5
"xyzzy"
7
```

the input:

```
pop stk
```

removes the top value from the stack and displays the resulting stack:

```
5
"xyzzy"
```

- **clrstk** This command clears the value stack. Remember that the contents of the stack are saved when you exit and re-enter KDB.

Arithmetic Operations

You can perform arithmetic operations on the top values on the stack:

+	compute [TOS-1] + [TOS]; pop 2; push result
-	compute [TOS-1] - [TOS]; pop 2; push result
*	compute [TOS-1] * [TOS]; pop 2; push result
/	compute [TOS-1] / [TOS]; pop 2; push result
%	compute [TOS-1] % [TOS]; pop 2; push result
>>	compute [TOS-1] >> [TOS]; pop 2; push result
<<	compute [TOS-1] << [TOS]; pop 2; push result
<	compute [TOS-1] < [TOS]; pop 2; push result
>	compute [TOS-1] > [TOS]; pop 2; push result
==	compute [TOS-1] == [TOS]; pop 2; push result
!=	compute [TOS-1] != [TOS]; pop 2; push result
&	compute [TOS-1] & [TOS]; pop 2; push result
|	compute [TOS-1] | [TOS]; pop 2; push result
^	compute [TOS-1] ^ [TOS]; pop 2; push result
&&	compute [TOS-1] && [TOS]; pop 2; push result
||	compute [TOS-1] || [TOS]; pop 2; push result
!	replace [TOS] with ![TOS]
++	replace [TOS] with [TOS] + 1
--	replace [TOS] with [TOS] - 1

For example, this input (subtracting 5 from 7):

```
7 5 - p
```

would produce this output:

```
2
```

The power of KDB's calculator feature lies in its ability to evaluate expressions like this:

```
callout 16 +
```

This pushes the address of the callout table on the stack and adds 16 to it. If the size of a callout table entry is 16 bytes, the result of the calculation is the address

of the second entry in the callout table. (Use the `size` command of `crash`(1M) to find the sizes of common system tables.)

WARNING: Make sure the divide operator (slash character) is both preceded and followed by spaces. If any other character appears next to the slash, it indicates a suffix instead of division.

READING AND WRITING TO MEMORY

These commands still operate like an RPN calculator, but they perform specific debugging operations instead of calculations. To examine and set the contents of memory (and I/O) use the commands:

`r`	replace [TOS] with the value at virtual address [TOS]
`w`	write [TOS-1] into virtual address [TOS]; pop 2
`dump`	show [TOS] bytes starting at virtual address [TOS-1]; pop 2

- `r` For example, you can find the value of the (long) kernel variable, `lbolt`, by typing:

  ```
  lbolt r p
  ```

  This puts the virtual address of `lbolt` on the stack, replaces it with the value found at that address, and prints the result.

- `w` To change the value of `lbolt` to 2000, type:

  ```
  2000 lbolt w
  ```

  This writes 2000 at `lbolt`'s virtual address.

  You could increment `lbolt` by typing:

  ```
  lbolt r ++ lbolt w
  ```

  This puts the virtual address of `lbolt` on the stack, replaces it with the value found at that address, adds 1 to the value, and writes the result at `lbolt`'s virtual address.

- `dump` This command displays a range of memory, both in hex and ASCII. For example, if you typed:

  ```
  putbuf 10 dump
  ```

  which shows 10 bytes, starting at the virtual address of `putbuf`, you would see something like:

  ```
  ........ ........ ........ 61746F74  D0108C50  ............tota
  6572206C 6D206C61 726F6D65 ........  D0108C60  l real memor....
  ```

  In each line, the block of four values on the left shows the values of 16 bytes, displayed as four 4-byte longwords in hex. The dots represent values outside of the requested range. (`dump` may also display question marks here; that means the address is invalid). The next column is the address of the first of the 16 bytes. The last column is the same 16 bytes displayed in ASCII. Dots here represent values outside the requested range or unprintable characters.

Suffixes

Suffixes can be appended to many KDB commands. They always begin with the slash character (/).

WARNING: Don't leave spaces before or after the slash character. When the slash is preceded and followed by a space, it indicates division instead of a suffix.

Operand-size suffixes. The r, w and dump commands can also work with units of bytes and words, as well as the default longs. To do this, append one of these suffixes to the command:

/b	byte
/w	word (2 bytes)
/l	long (4 bytes)—this is the default.

For example, to display the value of a short (2-byte) variable at address 0xD0008120, type:

```
0xD0008120 r/w p
```

Entering the dump command with /b displays 16 1-byte values per line, with /w displays eight 2-byte values per line, and with /l (or nothing) displays four 4-byte values per line.

Address-space suffixes. The r, w and dump commands, by default, work with kernel virtual addresses. You can change to physical addresses, I/O addresses, or user process virtual addresses by appending one of these suffixes to the command:

/k	kernel virtual — the default
/p	physical
/io	I/O port
/u#	user process number # virtual (# is a process slot number in hex)

- **/p** For example, to dump 40 (hex) bytes in longword format from physical address 2000, type:

  ```
  2000 40 dump/p
  ```

  The default address is kernel virtual, so the /p suffix is required for the physical address. Note that an operand-size suffix is not required, because long is the default.

- **/io** For example, to read from port 300 (in bytes) and display the result, type:

  ```
  300 r/io/b p
  ```

- **/u#** For example, to dump 20 longwords from process 16's u area at an offset of 1000, type:

  ```
  1000 u + 20 dump/u16
  ```

Suffix formats. Address-space suffixes can be combined with operand-size suffixes; only the first slash is required. For example, to do the read from I/O port 300 shown above, any of these command lines is acceptable:

```
300 r/io/b
300 r/b/io
300 r/iob
300 r/bio
```

Suffixes can also be attached directly to an address as shorthand for "read and print." Thus, `2000 r/p p` can be shortened to `2000/p`.

Since the default address-space is kernel virtual, the common operation of "read and print from kernel virtual" can be even further shortened. Type `lbolt/` to read and print the value of the (long) kernel variable, `lbolt`.

DISPLAYING AND WRITING TO REGISTERS

You can examine the CPU's general registers (and a couple of pseudo-registers) with these commands:

`%eax`	push the contents of 32-bit register `eax`
`%ebx`	push the contents of 32-bit register `ebx`
`%ecx`	push the contents of 32-bit register `ecx`
`%edx`	push the contents of 32-bit register `edx`
`%esi`	push the contents of 32-bit register `esi`
`%edi`	push the contents of 32-bit register `edi`
`%ebp`	push the contents of 32-bit register `ebp`
`%esp`	push the contents of 32-bit register `esp`
`%eip`	push the contents of 32-bit register `eip`
`%efl`	push the contents of 32-bit register `efl`
`%cs`	push the contents of 16-bit register `cs`
`%ds`	push the contents of 16-bit register `ds`
`%es`	push the contents of 16-bit register `es`
`%fs`	push the contents of 16-bit register `fs`
`%gs`	push the contents of 16-bit register `gs`
`%err`	push the error number
`%trap`	push the trap number
`%ax`	push the contents of 16-bit register `ax`
`%bx`	push the contents of 16-bit register `bx`
`%cx`	push the contents of 16-bit register `cx`
`%dx`	push the contents of 16-bit register `dx`
`%si`	push the contents of 16-bit register `si`
`%di`	push the contents of 16-bit register `di`
`%bp`	push the contents of 16-bit register `bp`
`%sp`	push the contents of 16-bit register `sp`
`%ip`	push the contents of 16-bit register `ip`
`%fl`	push the contents of 16-bit register `fl`
`%al`	push the contents of 8-bit register `al`
`%ah`	push the contents of 8-bit register `ah`
`%bl`	push the contents of 8-bit register `bl`
`%bh`	push the contents of 8-bit register `bh`
`%cl`	push the contents of 8-bit register `cl`

%ch	push the contents of 8-bit register ch
%dl	push the contents of 8-bit register dl
%dh	push the contents of 8-bit register dh

You can modify the values of general-purpose registers with these commands:

w%eax	write [TOS] into 32-bit register eax; pop 1
w%ebx	write [TOS] into 32-bit register ebx; pop 1
w%ecx	write [TOS] into 32-bit register ecx; pop 1
w%edx	write [TOS] into 32-bit register edx; pop 1
w%esi	write [TOS] into 32-bit register esi; pop 1
w%edi	write [TOS] into 32-bit register edi; pop 1
w%ebp	write [TOS] into 32-bit register ebp; pop 1
w%esp	write [TOS] into 32-bit register esp; pop 1
w%eip	write [TOS] into 32-bit register eip; pop 1
w%efl	write [TOS] into 32-bit register efl; pop 1
w%cs	write [TOS] into 16-bit register cs; pop 1
w%ds	write [TOS] into 16-bit register ds; pop 1
w%es	write [TOS] into 16-bit register es; pop 1
w%fs	write [TOS] into 16-bit register fs; pop 1
w%gs	write [TOS] into 16-bit register gs; pop 1
w%err	write [TOS] into the error number pseudo-register; pop 1
w%trap	write [TOS] into the trap number pseudo-register; pop 1
w%ax	write [TOS] into 16-bit register ax; pop 1
w%bx	write [TOS] into 16-bit register bx; pop 1
w%cx	write [TOS] into 16-bit register cx; pop 1
w%dx	write [TOS] into 16-bit register dx; pop 1
w%si	write [TOS] into 16-bit register si; pop 1
w%di	write [TOS] into 16-bit register di; pop 1
w%bp	write [TOS] into 16-bit register bp; pop 1
w%sp	write [TOS] into 16-bit register sp; pop 1
w%ip	write [TOS] into 16-bit register ip; pop 1
w%fl	write [TOS] into 16-bit register fl; pop 1
w%al	write [TOS] into 8-bit register al; pop 1
w%ah	write [TOS] into 8-bit register ah; pop 1
w%bl	write [TOS] into 8-bit register bl; pop 1
w%bh	write [TOS] into 8-bit register bh; pop 1
w%cl	write [TOS] into 8-bit register cl; pop 1
w%ch	write [TOS] into 8-bit register ch; pop 1
w%dl	write [TOS] into 8-bit register dl; pop 1
w%dh	write [TOS] into 8-bit register dh; pop 1

Register Sets

The commands listed above can also be used to access register sets. Multiple sets of general registers may be saved on the stack (one for each interrupt, trap, etc.). For more information see "Printing Kernel Stack Traces."

Register sets are numbered from 0 to 19, with 0 being the current (most recent) set. By default, the general-register commands use register set 0, but you can override this with a *register-set suffix*:

`/rs#`	register set number #

Note that by combining suffixes, you can access any register of any process. For example, you can get the eax register from process 5's register set 1 by typing:

```
%eax/u5rs1
```

to push the contents of that register (`%eax`) in register set 1 (`/rs1`) of user process 5 (`/u5`).

CPU Control Registers

In addition to the general registers, you can examine the values of CPU control registers with these commands:

`cr0`	push the contents of register `cr0`
`cr2`	push the contents of register `cr2`
`cr3`	push the contents of register `cr3`

CREATING DEBUGGER VARIABLES

KDB allows you to create named variables that are stored in the debugger and hold debugger values (numbers or strings). Two KDB commands apply to variables:

`=` *variable*	store [TOS] in [variable]; pop 1
`vars`	show values of debugger variables

- **= variable** This command assigns a value to a debugger variable. For example:

  ```
  5 = abc
  ```

  creates the variable `abc` if it does not exist, and sets the variable equal to 5. Now whenever you use the variable name, its value is pushed onto the stack. For example:

  ```
  abc abc + 2 - p
  ```

  (5 + 5 - 2) will yield `8`.

  Note that variable names share the same namespace as debugger macros and kernel global symbols.

- **vars** To look at all the existing variables, use the `vars` command. Variables are shown in the following format:

  name = *value*

  The `vars` command also lists macros, in this format:

  name `::` *value*

SETTING BREAKPOINTS

Set and modify breakpoints with these commands:

B	set breakpoint #[TOS] at address [TOS-1]; pop 2
-or-	set breakpoint #[TOS] at address [TOS-2] with command string [TOS-1]; pop 3
b	set first free breakpoint address [TOS]; pop 1
-or-	set first free breakpoint at address [TOS-1] with command string [TOS]; pop 2
brkoff	disable breakpoint #[TOS]; pop 1
brkon	re-enable breakpoint #[TOS]; pop 1
brksoff	disable all breakpoints
brkson	re-enable all (disabled) breakpoints
trace	set breakpoint #[TOS] trace count to [TOS-1]; pop 2
clrbrk	clear breakpoint #[TOS]; pop 1
clrbrks	clear all breakpoints
?brk	show current breakpoint settings

You can have up to four breakpoints, numbered 0 through 3, set at one time.

- **B and b** The `B` command lets you set specific breakpoints, while the `b` command automatically picks the first un-set breakpoint.

  This example sets breakpoint 3 at a specific address:

  ```
  0xD0125098 3 B
  ```

  Normally, you'll just set a breakpoint at a certain address. For example:

  ```
  read b
  ```

  This sets an instruction breakpoint at the beginning of the kernel `read` routine, using the next available breakpoint number. When the specified address is executed (after exiting from the debugger), you enter the debugger again, with a message indicating which breakpoint was triggered.

  Debugger command strings can be added to the breakpoint commands. Enter a quoted string of commands after the address:

  ```
  read "stack" b
  ```

  which is used as a series of debugger commands that are executed when the breakpoint is triggered. If there are several items in the string, separate them with spaces:

  ```
  ie6unitdata_req "300 r/bio p" b
  ```

  After these commands are executed, you are prompted for debugger commands, as usual, unless the `q` (quit) command is executed in the command string.

  You can append breakpoint-type suffixes to the breakpoint commands (`B` and `b`). By default, breakpoints are "instruction" breakpoints, which trigger when

the specified address is executed. The suffixes cause breakpoints to trigger on data accesses instead. The breakpoint-type suffixes are:

/a	data access breakpoint
/m	data modify breakpoint
/i	instruction execution breakpoint—this is the default

With access and modify breakpoints, you can also use operand-size suffixes to control the size of the address range that will trigger the breakpoint. The default is /1 (4 bytes); you can also use /w (word) and /b (byte). (See the earlier discussion of suffixes under "Reading and Writing to Memory" for more information.)

- **brkoff and brkon** These commands let you temporarily disable and re-enable a breakpoint, instead of clearing it with `clrbrk` and then re-entering it later. This is especially handy for breakpoints with command strings.
- **trace** This command sets a trace count for a breakpoint. This causes the debugger to just print a message and decrement the count when the breakpoint is triggered, instead of entering the debugger, until the count reaches zero. Commands attached to the breakpoint are not executed.
- **?brk** Use this command to determine the current breakpoint settings. Each set breakpoint is displayed, with (1) the breakpoint number, the address (both (2) in hex and (3) symbolic), (4) the current state, and (5) the type:

```
0: 0xD003907C(read) ON /i
1      2          3   4   5
```

The possible states are:

ON	set and enabled
DISABLED	set, but currently disabled
OFF	un-set (these breakpoints are not displayed by ?brk)

The possible types (in this example /i) are the same as the breakpoint-type suffixes described earlier.

If a breakpoint has a non-zero trace count, that is displayed after the breakpoint state. If a breakpoint has a command string, it is displayed at the end of the line. For example, with a count of 5 and a `stack` command, the above breakpoint would display as:

```
0: 0xD003907C(read) ON  0x5 /i "stack"
```

SINGLE-STEPPING THROUGH INSTRUCTIONS

The single-stepping commands do not work when you get into KDB through the kdb(1) command. However, if you access KDB through Ctrl-Alt-d, the `sysi86(SI86TODEMON)` system call, or enter KDB automatically, you can use these commands for single-stepping:

s	single step 1 instruction
ss	single step [TOS] instructions; pop 1
S	single step 1 instruction (passing calls)
SS	single step [TOS] instructions (passing calls); pop 1

s and ss single-step all instructions. S and SS single-step all instructions except call instructions. They don't step down into the called routine, but instead skip ahead to the return from the call, treating the whole subroutine sequence as a single instruction.

EXAMINING KERNEL DATA STRUCTURES

KDB provides commands for looking at certain kernel structures:

ps	show process information
sleeping	show list of sleeping processes
pinode	print s5 inode at address [TOS]; pop 1
puinode	print ufs inode at address [TOS]; pop 1
pprnode	print /proc inode at address [TOS]; pop 1
psnode	print snode at address [TOS]; pop 1
pvfs	print vfs struct at address [TOS]; pop 1
pvnode	print vnode at address [TOS]; pop 1

The sleeping command shows sleeping processes with their process table slot numbers and the channels on which they are waiting. This information can be used with the call and pstack commands.

PRINTING KERNEL STACK TRACES

KDB provides the following commands to look at kernel stack traces:

stack	kernel stack trace for the current process
pstack	kernel stack trace for process [TOS]; pop 1
stackargs	set max # arguments in stack trace to [TOS]; pop 1
stackdump	show contents of kernel stack in hex

Note that the argument to pstack can be specified either as a process table slot number, the address of the process structure, or -1 for the current process. (-1 pstack is equivalent to the stack command.)

The output of stack and pstack have the same format. A typical stack trace (for the current process, entered via <Ctrl-Alt-d>) looks like this:

```
DEBUGGER ENTERED FROM USER REQUEST
 kdcksysrq(D101FD40 D00DE624 81).....ebp:E0000D30  ret:D008F592
*kdintr+0x186(1 0).....ebp:E0000D74  ret:D0011A3A
INTERRUPT 0x1 from 158:D001218A (ebp:E0000D84)
   eax:      8 ebx:      0 ecx:FFFFFFFF edx:      8 efl:      246 ds: 160
  esi:D00EDDD0 edi:D106BC00 esp:E0000DC8 ebp:E0000DE0 regset:0 es:160
 idle(0 D00EDDD0 D106BC00).....(ebp:E0000DC4) ret:D006F11F
 pswtch(D002464C 0 D00F9090).....ebp:E0000DE0  ret:D00122ED
 swtch(0 D00F9090 D101A160).....(ebp:E0000DE4) ret:D002464C
 sleep(D0038B0C 14 D00BCA3C).....ebp:E0000DFC  ret:D0038D6F
 fsflush(0 E0000002 E0000002).....ebp:E0000E38  ret:D001E24B
 main+0x5FB().....ebp:E0000E70
```

The stack trace shows a history of which routine called which other routine, until the debugger was entered (or in the case of a non-current process, until the process was context-switched out).

The most-recently-entered routine is shown on the first line. In the example, the debugger was entered from `kdcksysrq`, which, in turn, was called by `kdintr`; `idle` was called from `pswtch`, and so on. The stack trace ends at the point the kernel was entered from user mode. In the case of a system process (as shown here) where there is no user mode, the stack trace ends at the call from `main`.

Routine Trace Format

The trace for each routine has four parts: (1) its address, (2) the arguments passed to it, (3) the value of its ebp register, and (4) its return address. For example:

```
fsflush(0 E0000002 E0000002).............ebp:E0000E38  ret:D001E24B
   1   ————————2————————                  ———3———  ———4———
```

Address. The address that was called usually appears in symbolic form. A routine name may also include:

An offset (a plus sign (+) and a hex number): `*kdintr+0x186`
The offset may mean that the actual address called was somewhere past the start of the indicated routine. This will most likely happen if a subroutine was declared "static." Since the debugger only has access to global symbols, it finds the nearest preceding global symbol.

The offset may also mean that the exact address called cannot be determined. The address displayed in this case is the return address into this routine from the routine *it* called. This will most likely happen if this routine was called indirectly via a function pointer.

An asterisk (*): `*kdintr+0x186`
This means the routine was called indirectly. There is insufficient information in a stack trace to be 100% sure of the correctness of indirect call traces.

A tilde (~)
This is used where there is some uncertainty in the stack trace that did not arise from indirect calls.

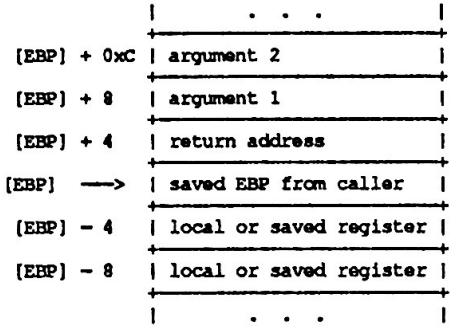
Whenever you see an asterisk or a tilde in a stack trace, there is a small chance that some part of the stack trace from that point on is incorrect.

Arguments. The arguments passed to the routine appear as a list of hex numbers, enclosed in parentheses. Since the actual number of arguments passed cannot be determined, KDB assumes that each routine has no more than a certain maximum number of arguments. The default is three, but you can change it with the `stackargs` command. If a routine actually has:

Fewer arguments than displayed:
Only the first ones are real. In rare cases when the debugger can deduce that a routine could not have been called with the maximum number of arguments (because there isn't enough room on the stack), it displays only the maximum possible number of arguments. In the above stack trace, the call to `kdintr` is shown with only two arguments `(1 0)`.

More arguments than displayed:
Increase the number with `stackargs` and then display the stack trace again, or dump out a portion of the stack directly in order to see all the arguments (continue to the next section for details).

ebp register. The value of the ebp register inside the routine is shown as a hex number following ebp:. This value can be used as a "frame pointer" to access arguments and local variables for the routine. The following diagram illustrates the stack layout.



For example, if you want to see all the arguments to a routine that takes five arguments, find its ebp value from the stack trace — say 0xE0000E0C — and enter these commands:

```
0xE0000E0C 8 + 5 4 * dump
```

or, more succinctly:

```
0xE0000E14 14 dump
```

Any ebp value in parentheses is a computed value (see the ebp values for idle and switch in the example). In these cases, due to code optimization or partial execution, the ebp value has not been set up for one or more routines. KDB computes the value ebp *ought to* have had and displays it in parentheses.

Return address. This is the address this routine returns to its caller. It is shown as a hex number following ret:.

Trap Frames

In addition to lines for each routine, stack traces will often include "trap frames" created when an event causes suspension of current processing, saving all register values on the stack. Typical events are interrupts, hardware exceptions, and system calls. Trap frames are three lines each, starting with an upper-case, non-indented keyword (like INTERRUPT in the example). The next two lines contain the values of the registers at the time the event occurred. The first line of a trap frame is in one of these formats:

```
INTERRUPT 0x1 from 158:D001218A (ebp:E0000D84)
TRAP 0x1(err 0x0)from 158:D001218A (ebp:E0000D94,ss:esp:1F:80468E8)
SYSTEM CALL from 158:D001218A (ebp: E0000D94, ss:esp: 1F:80468E8)
SIGNAL RETURN from 158:D001218A (ebp: E0000D94, ss:esp: 1F:80468E8)
```

These represent interrupts, hardware exception traps, system calls, and returns from old-style signal handlers, respectively. The number after INTERRUPT is the interrupt vector number (IRQ). The number after TRAP is the hardware exception number; the most common are 0x1 for breakpoint traps and 0xE for page faults.

The colon-separated numbers after the word from are the segment and offset (cs and eip) at the time the event occurred. The values in parentheses show the ebp value for the beginning of the trap frame, and the user stack pointer segment and offset at the time the event occurred. The user stack information is only displayed if the trap frame is for an entry into the kernel from user mode.

RESETTING THE NUMERIC BASE

If you don't start numbers with "0o" (for octal) or "0x" (for hex), KDB assumes they are in the default numeric base. Initially, the defaults for both input and output are set to 16 (hex), but you can use these commands to change them:

ibase	set default input base to [TOS]; pop 1
ibinary	set default input base to 2
ioctal	set default input base to 8
idecimal	set default input base to 10
ihex	set default input base to 16
obase	set output base to [TOS]; pop 1
ooctal	set output base to 8
odecimal	set output base to 10
ohex	set output base to 16

CONVERTING ADDRESS SPACES

Use these commands to convert another type of address to a physical address:

kvtop	convert kernel virtual address [TOS] to physical
uvtop	convert user proc #[TOS] address [TOS-1] to physical; pop 1

DOING CONDITIONAL EXECUTION

KDB provides two commands for conditional execution:

then	if [TOS] = 0, skip to endif; pop 1
endif	end scope of then command

In other words, a sequence like:

```
<condition> then <commands> endif
```

executes <commands> if and only if the <condition> is true (non-zero).

These are mostly useful for macros and breakpoint command strings. For example, imagine you wish to set a breakpoint for when the function inb is called with 2E as its first argument. Use the following command:

```
inb "%esp 4 + r 2E != then q" b
```

This says to set a breakpoint at inb, but enter the debugger only if the contents of (%esp+4) are equal to 2E. This works because esp points to the return address on the stack, and the longword after that is the first argument. For the second argument, you would add 8 instead of 4 (see the "Printing Kernel Stack Traces" section for details of the stack layout).

If you do a ?brk command, the display for that breakpoint includes the string of debugger commands:

```
0: 0xD003907C(inb) ON  /i "%esp 4 + r 2E != then q"
```

CALLING A KERNEL FUNCTION

Use this command to call an arbitrary kernel function:

`call`	call the function at address [TOS-1] with [TOS] arguments, given by [TOS-([TOS]+1)], ... [TOS-2]; pop [TOS]+2

To call psignal() with two arguments, the current process and 9, type:

```
curproc r 9 psignal 2 call
```

`curproc r` gives the value of the current process, the first argument, and 9 is the second argument. `psignal` is converted into the address at which that function can be called, and 2 specifies the number of arguments to pass to `psignal()`.

DOING A SYSTEM DUMP

This command causes a system dump and forces a reboot:

`sysdump`	cause a system dump

All of memory and the current state is dumped to the dump partition on the disk, so you can use `crash`(1M) to do a postmortem.

MISCELLANEOUS COMMANDS

Some miscellaneous KDB commands are:

`findsym`	print kernel symbol with address closest to [TOS]; pop 1
`dis`	disassemble [TOS] instructions starting at address [TOS-1]; pop 2
`nonverbose`	turn verbose mode off
`verbose`	turn verbose mode on
`newdebug`	switch to another debugger on next debugger entry
`help`	print a help message
`?`	print a help message (same as `help`)
`cmds`	print a list of all debugger commands

WRITING MACROS

KDB provides the ability to assign a string of commands to a single new command name, called a macro. When a debugging task involves repeating the same set of commands many times (possibly doing other things in between), it is easier to define a macro and use it in place of the whole set of commands.

These commands are used for macros:

`:: macro`	define [macro] as command string [TOS]; pop 1
`P`	print [TOS] in raw form; pop 1
`PP`	print [TOS] values in raw form, from [TOS-[TOS]], ... [TOS-1]; pop [TOS]+1
`vars`	show values of debugger macros and variables

- **:: macro** Use this command to define macros. For example:

```
"curproc r 16 - p" :: newaddr
```

  Note that macro names share the same namespace as debugger variables and kernel global symbols.

- **P and PP** These commands are provided to aid in writing macros. `P` and `PP` print values in *raw form*, without the embellishments provided by the `p` command, such as quotes around strings and automatic newlines after each value. This allows complete control over formatting. For example, the input:

```
"The value of curproc is " curproc r ".\n" 3 PP
```

  might produce the output:

```
The value of curproc is 0xD1011E80.
```

  To put something like this into a macro means putting strings inside strings, so you'll have to escape the inner quotes:

```
"\"The value of curproc is \" curproc r \".\n\" 3 PP" :: pcurproc
```

- **vars** Use this command to show the macro definitions. Macros are shown in this format:

  name :: *value*

  Note that the `vars` command also shows the values of variables, in this format:

  name = *value*

EXECUTING DEBUGGER COMMANDS AT BOOT TIME

KDB allows you to specify an arbitrary command sequence to be executed at boot time, when the system is coming up (specifically, from `main()` at the time of the `io_start` routines). You can do this by writing the commands into the file `$ROOT/etc/conf/cf.d/kdb.rc`, then rebuilding the kernel with *idbuild*.

Instead of rebuilding the kernel with *idbuild*, you can modify the KDB information in an already-built kernel by typing the command:

```
unixsyms -i /etc/conf/cf.d/kdb.rc /unix
```

At boot time, after the (possibly blank) string is executed, the system enters KDB at the `kdb>>` prompt, unless a `q` command was executed as part of the string — just like conditional breakpoints. (A non-existent or zero-length `kdb.rc` file acts as a single `q` command, so KDB is not entered.)

USING A SERIAL TERMINAL

KDB can be used from a serial terminal as well as the console. This is particularly useful if you are trying to debug a scenario that involves graphics or multiple virtual terminals on the console.

Before you attempt to use the debugger from a serial terminal, make sure there is a *getty* or *ttymon* running on it. It may be either logged in or waiting at the login prompt. This ensures that the baud rate and other parameters are properly set.

You can switch from the console to a terminal, and vice-versa, with the newterm command. This immediately switches you to the new terminal. The debugger continues to use this terminal until you give it the newterm command again, even if you exit and re-enter KDB.

The newterm command does not take an argument. On a 386, the serial terminal is assumed to be *tty00*, the terminal on the com1 port. You can change the device name by editing the `/etc/conf/pack.d/kdb-util/space.c` file, rebuilding the kernel and rebooting. If the terminal is attached to the com2 port, set the device name to *tty01* by changing all occurences of `asyputchar` and `asygetchar` to `asyputchar2` and `asygetchar2`, respectively, and changing the minor number of the device from 0 to 1. The first lines of 386-specific code should look like this:

```
#ifdef AT386
int asyputchar2(), asygetchar2();
static struct conssw asysw = {
     asyputchar2,     1,     asygetchar2
};
#endif
```

To use terminals on both com1 and com2 ports, you can set up newterm to cycle from the console to *tty00* to *tty01* and back to the console. Edit all the 386-specific code in the `space.c` file to look like this:

```
#ifdef AT386
int asyputchar(), asygetchar();
int asyputchar2(), asygetchar2();
static struct conssw asysw = {
     asyputchar,      0,     asygetchar
};
static struct conssw asysw2 = {
     asyputchar2,     1,     asygetchar2
};
#endif
     .
     .
     .

#ifdef AT386
     &asysw,
     &asysw2,
#endif
```

Once you exit from KDB, you can invoke it again from either the console or a serial terminal. Use the kdb command to invoke the debugger from a terminal; <Ctrl-Alt-d> only works from the console. Regardless of where you invoke KDB, its I/O appears where you directed it during the last KDB session.

ENTERING THE DEBUGGER FROM A DRIVER

If you are debugging a device driver or another part of the kernel, you can directly invoke the kernel debugger by including this code in your driver:

```
#include <sys/xdebug.h>

(*cdebugger) (DR_OTHER, NO_FRAME);
```

DR_OTHER tells the debugger that the reason for entering is "other." See sys/xdebug.h for a list of other reason codes.

Note that this mechanism cannot be used for debugging early kernel startup code or driver *init* routines, since the debugger cannot be used until its *init* routine (kdb_init) has been called.

DISABLING THE <Ctrl-Alt-d> SEQUENCE

As a security feature, KDB can only be called from the console via <Ctrl-Alt-d> if the kdb_security flag was set to 0 when the kernel was built. To disable the <Ctrl-Alt-d> key sequence, reset the kdb_security flag by using /etc/conf/bin/idtune to change the KDBSECURITY tunable to 1. Note that the flag setting does not affect the kdb command.

COMMAND SUMMARY

+	compute [TOS-1] + [TOS]; pop 2; push result
-	compute [TOS-1] - [TOS]; pop 2; push result
*	compute [TOS-1] * [TOS]; pop 2; push result
/	compute [TOS-1] / [TOS]; pop 2; push result
%	compute [TOS-1] % [TOS]; pop 2; push result
>>	compute [TOS-1] >> [TOS]; pop 2; push result
<<	compute [TOS-1] << [TOS]; pop 2; push result
<	compute [TOS-1] < [TOS]; pop 2; push result
>	compute [TOS-1] > [TOS]; pop 2; push result
==	compute [TOS-1] == [TOS]; pop 2; push result
!=	compute [TOS-1] != [TOS]; pop 2; push result
&	compute [TOS-1] & [TOS]; pop 2; push result
|	compute [TOS-1] | [TOS]; pop 2; push result
^	compute [TOS-1] ^ [TOS]; pop 2; push result
&&	compute [TOS-1] && [TOS]; pop 2; push result
||	compute [TOS-1] || [TOS]; pop 2; push result
!	replace [TOS] with ![TOS]
++	replace [TOS] with [TOS] + 1
--	replace [TOS] with [TOS] - 1
%eax	push the contents of 32-bit register eax
%ebx	push the contents of 32-bit register ebx
%ecx	push the contents of 32-bit register ecx
%edx	push the contents of 32-bit register edx

%esi	push the contents of 32-bit register esi
%edi	push the contents of 32-bit register edi
%ebp	push the contents of 32-bit register ebp
%esp	push the contents of 32-bit register esp
%eip	push the contents of 32-bit register eip
%efl	push the contents of 32-bit register efl
%cs	push the contents of 16-bit register cs
%ds	push the contents of 16-bit register ds
%es	push the contents of 16-bit register es
%fs	push the contents of 16-bit register fs
%gs	push the contents of 16-bit register gs
%err	push the error number
%trap	push the trap number
%ax	push the contents of 16-bit register ax
%bx	push the contents of 16-bit register bx
%cx	push the contents of 16-bit register cx
%dx	push the contents of 16-bit register dx
%si	push the contents of 16-bit register si
%di	push the contents of 16-bit register di
%bp	push the contents of 16-bit register bp
%sp	push the contents of 16-bit register sp
%ip	push the contents of 16-bit register ip
%fl	push the contents of 16-bit register fl
%al	push the contents of 8-bit register al
%ah	push the contents of 8-bit register ah
%bl	push the contents of 8-bit register bl
%bh	push the contents of 8-bit register bh
%cl	push the contents of 8-bit register cl
%ch	push the contents of 8-bit register ch
%dl	push the contents of 8-bit register dl
%dh	push the contents of 8-bit register dh
= *variable*	store [TOS] in [variable]; pop 1
:: *macro*	define [macro] as command string [TOS]; pop 1
?	print a help message (same as help)
?brk	show current breakpoint settings
B	set breakpoint #[TOS] at address [TOS-1]; pop 2 -or- set brkpoint #[TOS] at address [TOS-2] w/command string [TOS-1]; pop 3
b	set 1st free breakpoint address [TOS]; pop 1 -or- set 1st free brkpoint at address [TOS-1] w/command string [TOS]; pop 2
brkoff	disable breakpoint #[TOS]; pop 1
brkon	re-enable breakpoint #[TOS]; pop 1
brksoff	disable all breakpoints
brkson	re-enable all (disabled) breakpoints
call	call the function at address [TOS-1] with [TOS] arguments, given by [TOS-([TOS]+1)], ... [TOS-2]; pop [TOS]+2

clrbrk	clear breakpoint #[TOS]; pop 1
clrbrks	clear all breakpoints
clrstk	pop all values
cmds	print a list of all debugger commands
cr0	push the contents of register cr0
cr2	push the contents of register cr2
cr3	push the contents of register cr3
dis	disassemble [TOS] instructions starting at address [TOS-1]; pop 2
dump	show [TOS] bytes starting at virtual address [TOS-1]; pop 2
dup	push [TOS]
endif	end scope of then command
findsym	print kernel symbol with address closest to [TOS]; pop 1
help	print a help message
ibase	set default input base to [TOS]; pop 1
ibinary	set default input base to 2
ioctal	set default input base to 8
idecimal	set default input base to 10
ihex	set default input base to 16
kvtop	convert kernel virtual addr [TOS] to physical
newterm	alternate debugger I/O between console and tty00
newdebug	switch to another debugger on next debugger entry
nonverbose	turn verbose mode off
obase	set output base to [TOS]; pop 1
odecimal	set output base to 10
ohex	set output base to 16
ooctal	set output base to 8
P	print [TOS] in raw form; pop 1
p	print [TOS]
PP	print [TOS] values in raw form, from [TOS-[TOS]], ... [TOS-1]; pop [TOS]+1
pinode	print s5 inode at address [TOS]; pop 1
pop	pop 1 value
pprnode	print /proc inode at address [TOS]; pop 1
psnode	print snode at address [TOS]; pop 1
ps	show process information
pstack	kernel stack trace for process [TOS]; pop 1
pvfs	print vfs struct at address [TOS]; pop 1
pvnode	print vnode at address [TOS]; pop 1
puinode	print ufs inode at address [TOS]; pop 1
q	quit—exit from the debugger
r	replace [TOS] with the value at virtual address [TOS]
S	single step 1 instruction (passing calls)
s	single step 1 instruction
sleeping	show list of sleeping processes
SS	single step [TOS] instructions (passing calls); pop 1
ss	single step [TOS] instructions; pop 1

stack	kernel stack trace for the current process
stackargs	set max # arguments in stack trace to [TOS]; pop 1
stackdump	show contents of kernel stack in hex
stk	print all values on the stack
sysdump	cause a system dump
then	if [TOS] != 0, skip to endif; pop 1
trace	set breakpoint #[TOS] trace count to [TOS-1]; pop 2
uvtop	convert user process #[TOS] address [TOS-1] to physical; pop 1
vars	show values of debugger variables
verbose	turn verbose mode on
w	write [TOS-1] into virtual address [TOS]; pop 2
w%eax	write [TOS] into 32-bit register eax; pop 1
w%ebx	write [TOS] into 32-bit register ebx; pop 1
w%ecx	write [TOS] into 32-bit register ecx; pop 1
w%edx	write [TOS] into 32-bit register edx; pop 1
w%esi	write [TOS] into 32-bit register esi; pop 1
w%edi	write [TOS] into 32-bit register edi; pop 1
w%ebp	write [TOS] into 32-bit register ebp; pop 1
w%esp	write [TOS] into 32-bit register esp; pop 1
w%eip	write [TOS] into 32-bit register eip; pop 1
w%efl	write [TOS] into 32-bit register efl; pop 1
w%cs	write [TOS] into 16-bit register cs; pop 1
w%ds	write [TOS] into 16-bit register ds; pop 1
w%es	write [TOS] into 16-bit register es; pop 1
w%fs	write [TOS] into 16-bit register fs; pop 1
w%gs	write [TOS] into 16-bit register gs; pop 1
w%err	write [TOS] into the error number pseudo-register; pop 1
w%trap	write [TOS] into the trap number pseudo-register; pop 1
w%ax	write [TOS] into 16-bit register ax; pop 1
w%bx	write [TOS] into 16-bit register bx; pop 1
w%cx	write [TOS] into 16-bit register cx; pop 1
w%dx	write [TOS] into 16-bit register dx; pop 1
w%si	write [TOS] into 16-bit register si; pop 1
w%di	write [TOS] into 16-bit register di; pop 1
w%bp	write [TOS] into 16-bit register bp; pop 1
w%sp	write [TOS] into 16-bit register sp; pop 1
w%ip	write [TOS] into 16-bit register ip; pop 1
w%fl	write [TOS] into 16-bit register fl; pop 1
w%al	write [TOS] into 8-bit register al; pop 1
w%ah	write [TOS] into 8-bit register ah; pop 1
w%bl	write [TOS] into 8-bit register bl; pop 1
w%bh	write [TOS] into 8-bit register bh; pop 1
w%cl	write [TOS] into 8-bit register cl; pop 1
w%ch	write [TOS] into 8-bit register ch; pop 1
w%dl	write [TOS] into 8-bit register dl; pop 1
w%dh	write [TOS] into 8-bit register dh; pop 1

Command Suffixes

Operand size

/b	byte
/w	word (2 bytes)
/l	long (4 bytes)—this is the default

Address space

/k	kernel virtual—this is the default
/p	physical
/io	I/O port
/u#	user process number # virtual

Register set

/rs#	register set number #

Breakpoint type

/a	data access breakpoint
/m	data modify breakpoint
/i	instruction execution breakpoint—this is the default

Old Commands

These commands from previous versions are supported as aliases to new commands:

Old Command	New Equivalent
r1	r/b
r2	r/w
r4	r/l
w1	w/b
w2	w/w
w4	w/l
rp1	r/b/p
rp2	r/w/p
rp4	r/l/p
wp1	w/b/p
wp2	w/w/p
wp4	w/l/p
rio1	r/b/io
rio2	r/w/io
rio4	r/l/io
wio1	w/b/io
wio2	w/w/io
wio4	w/l/io
.trap	%trap
trc0	0 trace
trc1	1 trace
trc2	2 trace
trc3	3 trace
db?	?brk

These old commands are supported:

.i	push breakpoint type: instruction
.a	push breakpoint type: access byte
.m	push breakpoint type: modify byte
.aw	push breakpoint type: access word
.mw	push breakpoint type: modify word
.al	push breakpoint type: access long
.ml	push breakpoint type: modify long
.clr	push breakpoint type: clear breakpoint
brk0	set breakpoint 0 to type [TOS] at address [TOS-1]; pop 2
brk1	set breakpoint 1 to type [TOS] at address [TOS-1]; pop 2
brk2	set breakpoint 2 to type [TOS] at address [TOS-1]; pop 2
brk3	set breakpoint 3 to type [TOS] at address [TOS-1]; pop 2

SEE ALSO

dc(1), crash(1M).
The discussion of the UNIX kernel in the *System Administrator's Guide.*

NOTES

You can't set the KDB character in this version of KDB, even though the online usage message for the kdb command reads:

kdb ["char" *ASCII_code*]